



# Blitz Identity Provider

*version 5.23*

**Complete guide**

**Aug 02, 2024**

# Contents

<b>1</b>	<b>Functional specification</b>	<b>2</b>
<b>2</b>	<b>Administration</b>	<b>9</b>
2.1	Deployment	9
2.1.1	Deployment architecture	9
2.1.2	System requirements	10
	Operating systems	10
	Minimum requirements	10
	Recommended requirements for cluster	11
2.1.3	General installation instructions	14
	Step 1. JDK	14
	Step 2. Memcached	14
	Step 3. DBMS	15
	Step 4. RabbitMQ	17
	Step 5. Blitz Identity Provider	17
	Step 6. Configuration files synchronization	21
	Step 7. Web Server	23
	Step 8. LDAP directory	24
2.1.4	Express instructions for various operating systems	25
	Limitations when using instructions	26
	Rocky Linux, AlmaLinux, Oracle Linux, RHEL	26
	Step 1. JDK	27
	Step 2. Memcached	27
	Step 3. PostgreSQL	27
	Step 4. RabbitMQ	30
	Step 5. 389 Directory Server	32
	Step 6. Nginx	33
	Step 7. Blitz Identity Provider	33
2.1.5	The first steps after installation	36
	Configure launch options for Blitz Identity Provider services	36
	Logging in to Admin console	38
	License key installation	39
	Administrator account management	40
	Restarting Blitz Identity Provider services	41
	Deleting files used for installation	41
2.2	Basic configuration	41
2.2.1	User account attributes	41
	What is an account attribute?	41
	Configuring the available attributes	42
	Stored attributes	42
	Computed attributes	43
	Input value conversion rules	44
	Output value conversion rules	45
	Setting up attribute purpose	45
	Connecting attribute storages	46
	Types of storage	46

	Connecting storage via LDAP . . . . .	47
	Connecting to storage via REST . . . . .	51
	Configuring internal storage . . . . .	57
2.2.2	Authentication . . . . .	58
	How to work with authentication settings . . . . .	58
	General settings . . . . .	60
	Password policies . . . . .	62
	Security key management . . . . .	63
	Configuring security keys . . . . .	63
	Logging in via WebAuthn, Passkey, FIDO2 . . . . .	65
	Login confirmation with WebAuthn, Passkey, FIDO2, U2F . . . . .	67
	Logging in using login and password . . . . .	67
	Logging in with electronic signature tool . . . . .	71
	Configuring the authentication method in the Admin console . . . . .	71
	Using and updating the plug-in . . . . .	73
	Logging in via external identification services . . . . .	73
	Logging in with proxy authentication . . . . .	74
	Logging in using operating system session . . . . .	75
	Domain controller (Kerberos server) configuration . . . . .	76
	Settings in Blitz Identity Provider admin console . . . . .	78
	Users' browsers configuration . . . . .	79
	Blitz Identity Provider application launch settings . . . . .	81
	Web Server configurations . . . . .	81
	Debugging operating system session login problems . . . . .	82
	Logging in with email . . . . .	82
	Step 1. Add the method to blitz.conf . . . . .	82
	Step 2. Configure the method in the console . . . . .	83
	Logging in with confirmation codes . . . . .	84
	Logging in from known device . . . . .	86
	Logging in by one-time link . . . . .	86
	Logging in by QR code . . . . .	87
	Automatic user identification by session properties . . . . .	88
	Step 1. Create the login procedure . . . . .	88
	Step 2. Add a method to blitz.conf . . . . .	88
	Step 3. Configure the method in the console . . . . .	89
	Step 4. Customization of texts . . . . .	90
	Log-in confirmation with a HMAC-based one-time password (HOTP) . . . . .	91
	Time-based one-time password log-in confirmation (TOTP) . . . . .	92
	Binding devices to user accounts . . . . .	93
	Binding of hardware keyfobs . . . . .	93
	Binding a mobile application . . . . .	95
	Confirmation codes sent in SMS and push notifications . . . . .	96
	Confirmation codes sent by email . . . . .	98
	Log-in confirmation via Duo Mobile . . . . .	99
	Re-confirmation when logging in from known device . . . . .	102
	Confirmation by answering security question . . . . .	102
	Step 1. Add method to blitz.conf . . . . .	102
	Step 2. Create directory of security questions . . . . .	103
	Step 3. Configure method in console . . . . .	103
	Confirmation by incoming call . . . . .	104
	Step 1. Add the method to blitz.conf . . . . .	104
	Step 2. Configure the method in the console . . . . .	105
	Configuring an external authentication method . . . . .	108
	Customizing the Impersonalization Procedure . . . . .	109
2.2.3	External identity providers . . . . .	109
	How to set up login via external identity providers . . . . .	109
	International providers . . . . .	110
	Apple ID . . . . .	110

	Google . . . . .	115
	Facebook . . . . .	117
	Login via another Blitz Identity Provider setup . . . . .	119
	Account linking settings . . . . .	121
	Basic configuration . . . . .	122
	Advanced configuration . . . . .	124
2.2.4	Customizing user services . . . . .	126
	General settings . . . . .	126
	User registration . . . . .	128
	Registration form . . . . .	128
	Registration service settings . . . . .	130
	Registration procedure . . . . .	131
	Changing the text in the User agreement . . . . .	131
	User profile . . . . .	131
	Displaying user attributes . . . . .	132
	Additional parameters . . . . .	133
	Access recovery . . . . .	134
	Console settings . . . . .	134
	Form texts . . . . .	136
2.2.5	User administration . . . . .	137
	User account management . . . . .	137
	User search . . . . .	138
	Adding a user . . . . .	139
	View and edit user attributes . . . . .	139
	Editing attributes . . . . .	141
	Resetting sessions . . . . .	141
	Changing the password . . . . .	142
	View and unlink external providers . . . . .	142
	Binding devices for 2FA with a one-time password . . . . .	142
	Binding Duo Mobile . . . . .	144
	Group Membership Management . . . . .	144
	Viewing, assigning, and revoking rights . . . . .	145
	Memorized devices and browsers . . . . .	147
	Security keys . . . . .	148
	Permissions granted to applications . . . . .	148
	Managing user groups . . . . .	149
	Enabling the display of groups in blitz.conf . . . . .	149
	Working with groups . . . . .	150
	Access rights management . . . . .	151
2.2.6	Notifications and sending messages . . . . .	152
	Configuring connection to SMS gateway . . . . .	153
	Connection to the service of sending push notifications . . . . .	155
	Configuring the connection to the SMTP gateway . . . . .	156
2.3	Access to applications and network services . . . . .	157
2.3.1	Registering applications in Blitz Identity Provider . . . . .	157
	About applications . . . . .	157
	Creating a new application account . . . . .	158
2.3.2	Operation schemes of SSO technologies . . . . .	162
	Connecting a web app via OIDC . . . . .	162
	Connecting a mobile app via OIDC . . . . .	164
	Connecting an app via SAML . . . . .	166
2.3.3	Configuring SAML and WS-Federation . . . . .	167
	Connection via SAML 1.0/1.1/2.0 . . . . .	167
	Connection via WS-Federation . . . . .	168
	Uploading SAML metadata . . . . .	169
	Configuring SAML attribute . . . . .	170
2.3.4	OAuth 2.0 and OpenID Connect 1.0 . . . . .	171
	Configuring the application . . . . .	171



	General OAuth 2.0 settings . . . . .	175
	Adding attributes to an identity token . . . . .	177
	Configuring Dynamic OAuth 2.0 Client Registration . . . . .	179
2.3.5	Simple . . . . .	181
2.3.6	Interaction via the REST API . . . . .	184
2.3.7	Access to network services via RADIUS . . . . .	184
	Step 1. Configure the RADIUS Server . . . . .	185
	Step 2. Configure the application . . . . .	189
	Step 3. Configuration on the network service side . . . . .	190
2.4	Customization with Java code . . . . .	191
2.4.1	Login procedures and their creation . . . . .	191
	About the login procedures . . . . .	191
	Creating a procedure . . . . .	191
2.4.2	Ready-made login procedures . . . . .	193
	Forced two-factor authentication . . . . .	194
	Limiting the list of available first factor methods . . . . .	194
	Log in only with a certain attribute value . . . . .	195
	Prohibiting login after account expiration . . . . .	197
	Log in only from certain networks . . . . .	197
	Prohibition of work in several simultaneous sessions . . . . .	199
	Saving a list of user groups in claims . . . . .	199
	Displaying an announcement to the user at login . . . . .	200
	Procedure . . . . .	200
	Adding a procedure to blitz.conf . . . . .	202
	Request for user to enter attribute or actualize phone and email . . . . .	202
	Requesting the user to enter a security question . . . . .	205
	Registration of security key (WebAuthn, Passkey, FIDO2) at login . . . . .	206
	Display a list of value selections to the user at login . . . . .	208
	Procedure . . . . .	208
	Adding a procedure to blitz.conf . . . . .	209
2.4.3	Functions and methods of various purposes in login procedures . . . . .	210
	Obtaining the user's geodata . . . . .	210
	User session reset . . . . .	212
	Invoking custom errors in script . . . . .	213
	Analyzing application tags . . . . .	214
2.4.4	Customization of the logic of operations with data storages . . . . .	215
	Customization principle . . . . .	215
	Configuration . . . . .	216
	Writing a custom procedure . . . . .	216
2.4.5	Procedures for binding external user accounts . . . . .	217
	User registration in external identity provider . . . . .	219
	Discovering external account name . . . . .	220
2.5	Design and UI texts . . . . .	221
2.5.1	Login page . . . . .	221
	Editing the default template . . . . .	222
	Creating and modifying new templates using the constructor . . . . .	226
	Creating and modifying new templates in manual mode . . . . .	227
2.5.2	User profile . . . . .	230
	Header logo . . . . .	230
	Footer logo . . . . .	231
	Color scheme customization . . . . .	231
2.5.3	Multilanguage support . . . . .	231
2.5.4	Interface text settings . . . . .	234
	Web interface texts . . . . .	234
	Email and SMS templates . . . . .	234
	Device and browser names . . . . .	240
	Messages for different applications . . . . .	241
	Auxiliary application messages (pipes) . . . . .	241

2.5.5	Logos for external provider log-in buttons	243
2.6	Configuration file settings	244
2.6.1	Configuration file list	244
2.6.2	Settings in blitz.conf file	245
	Logins and passwords	246
	Number of password verifications	246
	Password change at login	246
	System names of login and password fields	246
	Attributes	247
	External attribute validator	247
	Attribute translator	248
	CAPTCHA	249
	Queue server	254
	Sending events to queue server	254
	Queue server as a message broker	255
	Stores and databases	257
	Storing objects in Couchbase	257
	Reading the Couchbase Server cluster configuration	257
	Object storage time	258
	Advanced PostgreSQL connection settings	258
	Advanced LDAP connection settings	259
	Geodatabase	261
	Several DBMSs usage	261
	Blitz Identity Provider domain	262
	Users	264
	Blocking inactive users	264
	Prohibit reuse of the remote user ID	265
	WebAuthn, Passkey, FIDO2, U2F provider certificates	265
	OIDC, SAML, and external identity providers	266
	OIDC Discovery service	266
	Call addresses of external providers	266
	External SAML provider	266
	Logging incomplete login attempts	269
	Transferring security events to file or Kafka	269
	Storing application settings in separate files	277
	SSO session duration	278
2.6.3	Admin console settings	279
	Logging in to admin console via SSO	279
	Session limit	281
	Roles and permissions for the console	281
	Changing console admin password	282
2.6.4	Configuring Token Exchange	282
	Step 1. Create service access rules	282
	Step 2. Configuring access token exchange	286
2.7	Security, maintenance, and troubleshooting	287
2.7.1	Viewing security events	287
2.7.2	Application performance monitoring	287
	Standard monitoring service	287
	Using Grafana and Prometheus	290
2.7.3	Problem solving	291
2.7.4	Security gateway	293
<b>3</b>	<b>Integration</b>	<b>294</b>
3.1	Preparing for integration	294
3.1.1	Selecting an interaction protocol	294
3.2	OIDC application integration	295
3.2.1	How to register the application correctly	295
3.2.2	Connecting a web application	299

	Connection settings . . . . .	299
	Ready-made libraries . . . . .	300
	Getting the authorization code . . . . .	300
	Getting tokens . . . . .	304
	ID token . . . . .	309
	Checking the access token through the introspection service . . . . .	312
	Verification of the access token by the application . . . . .	314
	Logout . . . . .	314
3.2.3	Connecting a mobile app . . . . .	317
	Connection settings . . . . .	317
	Ready-made libraries . . . . .	318
	Dynamic registration of an application instance . . . . .	318
	User's initial login . . . . .	320
	Getting the authorization code . . . . .	320
	Getting tokens by an application instance . . . . .	322
	User re-login . . . . .	323
	User switching or logging out . . . . .	324
	Opening web resources from the application . . . . .	324
	Login to the application using a QR code . . . . .	325
3.2.4	Connecting Smart Device (IoT) applications . . . . .	330
	General information . . . . .	330
	Connection settings . . . . .	330
	Getting the authorization code . . . . .	331
	Getting a security token . . . . .	332
3.2.5	Getting user attributes . . . . .	333
3.2.6	Ensuring connection security . . . . .	334
3.3	SAML application integration . . . . .	334
3.3.1	How to register the application correctly . . . . .	334
3.3.2	Connecting the application via SAML . . . . .	336
	Connection data . . . . .	336
	Ready-made libraries . . . . .	338
	Principle of integration . . . . .	339
	Identification and authentication . . . . .	339
	Logout . . . . .	339
3.4	User management API . . . . .	339
3.4.1	General information . . . . .	339
	REST API versions . . . . .	339
	REST API access modes . . . . .	340
	User access mode . . . . .	340
	System access mode . . . . .	342
3.4.2	Accounts . . . . .	346
	Registration . . . . .	347
	Search . . . . .	355
	Attributes . . . . .	357
	Getting attributes . . . . .	357
	Changing an attribute . . . . .	358
	Changing the phone number . . . . .	359
	Changing the email address . . . . .	362
	Passwords . . . . .	366
	Changing the password . . . . .	366
	Changing the password of subordinate account . . . . .	372
	Authentication modes . . . . .	373
	Checking the status . . . . .	373
	Changing authentication modes . . . . .	374
	User properties . . . . .	375
	Obtaining properties . . . . .	375
	Adding, modifying, and deleting properties . . . . .	376
	TOTP . . . . .	378

	Checking for TOTP availability . . . . .	378
	TOTP linking . . . . .	379
	Deleting the linking . . . . .	380
	Account status . . . . .	381
	Checking account status . . . . .	381
	Changing the account status . . . . .	382
	External providers . . . . .	383
	List of external providers . . . . .	383
	Linking a provider by ID . . . . .	383
	Linking a provider . . . . .	384
	Deleting a provider linking . . . . .	385
	Obtaining a user access token . . . . .	385
	Audit events . . . . .	386
	Known devices and sessions . . . . .	389
	List of known devices . . . . .	389
	Deleting a device from the list . . . . .	389
	Resetting user sessions . . . . .	390
	Security questions . . . . .	391
	Checking for a question . . . . .	391
	Checking the answer . . . . .	391
	Setting or changing a question . . . . .	392
	Deleting a question . . . . .	393
	Permissions issued by the user . . . . .	393
	List of permissions . . . . .	393
	Revocation of permission . . . . .	394
	Mobile apps . . . . .	394
	List of mobile apps . . . . .	394
	Unlinking from a mobile app account . . . . .	395
	Deleting an account . . . . .	395
3.4.3	User groups . . . . .	395
	Getting group attributes by id . . . . .	396
	Search for a group by attribute . . . . .	396
	Creating a group . . . . .	397
	Changing group attributes . . . . .	398
	Deleting a group . . . . .	399
	Getting a list of users in a group . . . . .	399
	Adding users . . . . .	401
	Removing users . . . . .	402
3.4.4	Access rights . . . . .	403
	List of user rights . . . . .	404
	List of application rights . . . . .	404
	Rights in relation to the user . . . . .	405
	Rights in relation to a group of users . . . . .	406
	Rights in relation to the application . . . . .	406
	Assignment of rights . . . . .	407
	Revocation of rights . . . . .	410
	The rights of the master user in relation to the slave . . . . .	413
3.5	Advanced features . . . . .	416
3.5.1	Additional authentication method . . . . .	416
	Request handler service . . . . .	416
	Transmission of the authentication result . . . . .	418
	Method verification service . . . . .	419
3.5.2	Invoking the auxiliary application at the moment of login . . . . .	419
	Request to open the application . . . . .	420
	Returning the user to Blitz Identity Provider . . . . .	420
3.5.3	Administration API . . . . .	421
	Getting application settings . . . . .	423
	Application registration . . . . .	425

	Changing application settings . . . . .	427
	Deleting an application . . . . .	429
3.5.4	Invoking a third-party user registration application . . . . .	430
	Registration Initiation Service . . . . .	430
	Registration completion service . . . . .	431
3.5.5	Authentication API . . . . .	432
	Settings for using the API . . . . .	433
	Interaction scheme . . . . .	433
	Starting the login process . . . . .	435
	Logging in using login and password . . . . .	437
	Login by phone and confirmation code . . . . .	442
	Logging in with email . . . . .	445
	Login by QR code . . . . .	446
	Confirmation of login by confirmation code . . . . .	448
<b>4</b>	<b>Modules</b>	<b>451</b>
4.1	Blitz Keeper security gateway . . . . .	451
4.1.1	About Blitz Keeper . . . . .	451
4.1.2	Installing the blitz-keeper service . . . . .	452
4.1.3	Configuring Blitz Keeper . . . . .	453
4.1.4	Creating service access rules . . . . .	454
4.1.5	Configuring access token exchange . . . . .	454
4.1.6	Viewing logs . . . . .	454
4.2	Blitz Panel app showcase . . . . .	454
4.2.1	About Blitz Panel . . . . .	454
4.2.2	Installing the blitz-panel service . . . . .	455
4.2.3	Blitz Panel configuration . . . . .	456
4.2.4	Blitz Panel design and localization . . . . .	463
	Appearance modification . . . . .	463
	Adding a language . . . . .	463
4.2.5	Viewing logs . . . . .	464

Blitz Identity Provider protects user accounts - providing out-of-the-box, flexible, customizable and best practice account protection features.

Blitz Identity Provider provides Internet users access to company websites and mobile applications, as well as employee access to internal company resources and cloud services.

**Key features of Blitz Identity Provider:**

- providing a single end-to-end user login to applications (Single Sign-On);
- two-factor authentication;
- configurable user interface of the login, registration, access recovery, account management pages;
- login using external identity providers: login using social network accounts, federated login using external identity providers;
- checking access rights for user logins to applications;
- verification of user and application access rights using REST-services;
- logging of access history and account activities.

# Chapter 1

## Functional specification

Functions group	Functions
Single Sign On Technologies	
OpenID Connect and OAuth 2.0	RFC 6749 “The OAuth 2.0 Authorization Framework” OpenID Connect Core 1.0 Sending user attributes as part of id_token/access_token into JSON Web Token (JWT) Configurable REST service UserInfo, customizable returned attributes depending on scope RFC 7636 “Proof Key for Code Exchange by OAuth Public Clients” RFC 7662 “OAuth 2.0 Token Introspection” RFC 7591 “OAuth 2.0 Dynamic Client Registration Protocol” RFC 7592 “OAuth 2.0 Dynamic Client Registration Management Protocol” RFC 8252 “OAuth 2.0 for Native Apps” RFC 8414 “OAuth 2.0 Authorization Server Metadata” OpenID Connect RP-Initiated Logout 1.0 OpenID Connect Front-Channel Logout 1.0 OpenID Connect Back-Channel Logout 1.0
SAML	SAML Web Browser SSO Profile SAML Single Logout Profile
RADIUS	RFC 2865 “Remote Authentication Dial In User Service (RADIUS)”
WS-Federation	WS-Federation (to connect Microsoft applications)
Proxy SSO	Connections of web applications receiving session status from HTTP headers and cookies Supports ability to send user account login/password to proxy hosted web application, that doesn't have default support for SSO connections
Other	Single Sign-On works between applications that are connected to IDP using any supported technology (for example, SSO between OpenID Connect and SAML applications) Supports SSO login using Kerberos SSO Supports SSO with IBM applications using Ltpa2Token for Single Sign-on
Identification and authentication	
Logging in using login and password	Login/password verification during authentication

continues on next page

Table 1 – continued from previous page

Functions group	Functions
	<p>Ability to use several entities (phone, email, login) as login simultaneously and enter login in different formats (e.g. phone as +7..., 8..., with different brackets, hyphens, spaces)</p> <p>Remembering login if user has logged in from device before</p> <p>Remembering multiple users on the device. Ability to change the current user account without having to logout</p> <p>Event handling “password must be changed” on login. Changing password during login</p> <p>Verifying password for compliance with existing password policy during login. Recommendation to change password</p> <p>Built-in protection against password brute force (trying to brute force passwords for one account) and login brute force (trying to brute force a password for a set of accounts):</p> <ul style="list-style-type: none"> <li>• CAPTCHA verification (reCAPTCHA or other service chosen by the Customer)</li> <li>• temporary blocking of login by account password in event of detecting brute force attempts</li> <li>• user login slowdown (login delay, browser solving a computationally complex task - Proof of Work)</li> </ul> <p>User notification when attempting to login with a recently changed password</p>
Logging in based on session	<p>User identification based on domain login (Kerberos)</p> <p>Capability to connect login simultaneously to multiple domains and provide end-to-end user login of from different domains</p> <p>Capability to configure that OS session-based login applies only to logins from internal networks and PCs, but not for mobile app logins and logins outside of the internal network</p>
Logging in via social network account/external identity provider	<p>Social networks and external identity providers that support log in of users without the need to edit or code connectors: Apple ID, Google, Facebook?</p> <p>Logging in via an external identity provider with OIDC support</p> <p>Logging in via an external identity provider with SAML support</p> <p>Account matching/registration during initial login via a social network</p> <p>Ability to bind multiple external provider accounts simultaneously to a single user account</p> <p>Ability to bind multiple user accounts simultaneously to single external provider account</p> <p>Ability to program your own algorithm for account binding and attribute matching</p> <p>Ability to store access tokens issued by external identity providers</p>

continues on next page



Table 1 – continued from previous page

Functions group	Functions
Logging in based on remembered device	Automated identification of the user if the he/she has logged in from that device before and agreed to remember the login
	Allows the user to track which devices have remembered their login and log out from those devices
	Automatic logout from remembered devices if user changes/recovers password
Automatic identification by session properties	Automatic identification of the user by session properties. All properties are supported. sessions that can be defined by the Customer and provided in Blitz Identity Provider. Flexible method configuration and full customization of interface texts.
Logging in via WebAuthn, Passkey, FIDO2	Logging in via platform-independent security keys FIDO2
	Logging in via platform-specific Passkey / FIDO2 security keys - Windows Hello (pin code, fingerprint), Passkey, password or Touch ID from MacBook, Passkey, Face ID or Touch ID of iOS or Android smartphone or tablet
Logging in via smart card / USB key	Logging in via qualified electronic signature
	Supported electronic signature tools: CryptoPro CSP 3.9 and higher, VipNet CSP 4.2, Signal-COM CSP 3.0, Rutoken, JaCarta, ISBC ESMART, SafeNet eToken
	Supported user OS: Windows 8.1/10/11, macOS 10.13/10.14/10.15/11/12/13, Linux Debian 9, Mint 19, Ubuntu 18, Astra Linux 1.7, Red OS 7.3
	Supported browsers: Internet Explorer 11, Chrome, Firefox
	Account matching/registration during initial login based on data from qualified electronic signature certificate
	Ability to verify signature/certificate validity using built-in software features
	Ability to verify signature/certificate validity via external verification service
Two-factor authentication	Login confirmation with one-time password sent by SMS (SMS-gateway is provided by Customer)
	Login confirmation with a one-time password from email
	Login confirmation with a one-time TOTP application password (RFC 6238 "TOTP: Time-Based One-Time Password Algorithm")
	Login confirmation with a one-time password from the hardware keyfob. Support for HOTP keyfobs (RFC 4226 "HOTP: An HMAC-Based One-Time Password Algorithm"). The keyfobs are provided by the customer
	Login confirmation with security key WebAuthn, Passkey, FIDO2
	Login confirmation with U2F security key
	Login confirmation by one-time password in push-notification in customer's mobile application (service for sending push-notifications and mobile application are provided by the Customer)
	Login confirmation with Flash Call

continues on next page

Table 1 – continued from previous page

Functions group	Functions
Other	Ability for a customer to add their own authentication method
	Ability for a customer to customize the appearance of the login page separately for each application
	Providing API, that allows mobile apps to register a login event and receive security tokens when using PIN, Touch ID and Face ID logins
	Blocking accounts in case of long inactivity
	Prohibition of deleted account ID reuse within a specified time
	Ability to analyze user geodata
Logout	
Logout	Ending user session when user logs out
	Ending user session when the user's password changes in another session, or when resetting/recovering the user's password
	Limitation on acceptable links to return to application after successful logout
	Application notification of a single logout via browser (front channel)
	Application notification of a single logout via server (back channel)
Access control	
Access control	Verifying access rules when a user logs into applications. Verifying user access rights, membership in user groups, attributes with required values
	Verifying access rules when applications call protected REST services via Blitz Keeper (API Security Gateway)
Account management	
Registration	Customizable self-registration web application. You can customize the set of attributes to be filled in by the user during registration, email/phone confirmation requirements, customize the appearance of the registration page, call the Customer's verification services
	You can configure different user self-service login web application settings for different scenarios of registration invoke
	The ability to invoke an external registration application and pass it the login information and data obtained from an external provider during the login process
	After successful registration, the user automatically logs in to the application, that originally initiated the registration procedure
	CAPTCHA verification (reCAPTCHA or other service chosen by the Customer)

continues on next page

Table 1 – continued from previous page

Functions group	Functions
Account security settings	<p>A web application that allows the user to self-manage his/her account security setting:</p> <ul style="list-style-type: none"> <li>• ability to change password;</li> <li>• ability to edit some attributes. Including the ability to edit phone number with confirmation via SMS code and the ability to edit email with confirmation via code/link via email;</li> <li>• ability to set up two-factor authentication for your user account;</li> <li>• ability to view/edit list of remembered devices, bound accounts of external login providers;</li> <li>• ability to view security events with your user account.</li> </ul> <p>Providing an API to be able to embed all of the above features to manage account security settings in the external web application</p>
Forgotten password recovery	A web application that allows to recover a forgotten password with email or mobile confirmation
	Additional checks during password recovery from an account for which two-factor authentication is enabled
	After successful password recovery, the user automatically logs in to the application, that originally initiated the recovery procedure
	CAPTCHA verification (reCAPTCHA or other service chosen by the Customer)
Account actions when login	Ability to set a phone number (if not present) in the account at login time or confirm phone relevance (if it is time to confirm relevance)
	Ability to set a phone number (if not present) in the account at login time or confirm phone relevance (if it is time to confirm relevance)
	Ability to set an email address (if missing) in the account at login time or to confirm the relevance of the email address (if it is time to confirm relevance)
	Ability to issue a Passkey at the moment of login (customize Face ID / Touch ID login)
	Ability to show the user an announcement
	Ability to request consent from the user
	Ability to request the user to fill in a text attribute
	Ability to ask a security question at the moment of login
	Ability to build your own business process of interaction with the user at login to the application (e.g., display an informational message to the user in some situations or request to lead something)
Password policies	Password verification for compliance with password policies: minimum length, alphabetical requirements, prohibition of dictionary passwords, no duplicate passwords, expiration validation
Advanced features	
Customization of the logic of work using Java programming	Setting user login rules for applications through login and registration procedures

continues on next page

Table 1 – continued from previous page

Functions group	Functions
	Customization of data storage operations
Monitoring and auditing	
Alerts users about security events	Notification of users of security events with their accounts: login from an unusual device, password change (changed it yourself, password reset by administrator, password reset due to password recovery), binding to a social network, enabling/disabling two-factor authentication Ability to configure notification events and notification texts for SMS and email
Security events logging	Logging of successful and unsuccessful security events with the account: login events, registration, change of security settings, password recovery. Both user-initiated and administrator-initiated actions should be logged Logging of successful and unsuccessful security events with the account: login events, registration, change of security settings, password recovery. Both user-initiated and administrator-initiated actions should be logged Matching IP-addresses to geodata in events and notifications (database in mmdb format with geodata is provided by the Customer) Administrators interface for searching/viewing security events Logging security events: to the database, to a log file, to Kafka
Monitoring	Ability to invoke metrics and statistics collection systems, antifraud systems at user login Ability to monitor components from external monitoring system (Zabbix and similar). Ability to provide Prometheus metrics Grafana dashboard templates and Prometheus job assignments are available
Queues	Ability to queue RabbitMQ events associated with user accounts and access groups Ability to send security events to Kafka
Administration	

continues on next page

Table 1 – continued from previous page

Functions group	Functions
Administration	Admin web application: <ul style="list-style-type: none"> <li>• configuring connected application settings (application parameters, allowed interaction modes, access control rules)</li> <li>• configuring user attributes and mapping attributes to an account store</li> <li>• configuring connection to LDAP-based account stores</li> <li>• configuring connection to random stores (service is provided by the Customer)</li> <li>• support of simultaneous connection to multiple account stores</li> <li>• configuring identity/authentication methods and external login providers</li> <li>• configuring the connection to SMTP service and SMS gateway</li> <li>• support for role-based access for logging into the administrator web application. Ability to set different actions available for different users</li> <li>• administration of web application registration settings, security settings, password recovery settings</li> <li>• user account administration (search, view, manage attributes, two-factor authentication settings, bindings of memorized devices and social networks, memorized user browsers, reset sessions, reset password, lock/unlock account, manage security keys, manage membership in user groups, assign/revoke access rights)</li> <li>• administration of user groups, management of user group memberships</li> <li>• configuring web applications login page themes</li> <li>• viewing and filtering of logged security events</li> <li>• ability to enter admin web application via SSO</li> </ul>
	Admin interface in English and Russian
	Ability to add additional languages

# Chapter 2

## Administration

### 2.1 Deployment

#### 2.1.1 Deployment architecture

The operation of Blitz Identity Provider is based on the interaction of the following architectural components:

1. Web Server. You can use your company's existing web server to load balance and remove SSL encryption from incoming traffic.
2. Blitz Identity Provider services:
  - `blitz-console` – admin console Blitz Console;
  - `blitz-idp` - authentication service and User profile;
  - `blitz-registration` – registration service;
  - `blitz-recovery` – access recovery service;
  - `blitz-keeper` – [security gateway](#) (page 451);
  - `blitz-panel` – a `:ref:panel <blitz-panel>` that provides users with quick access to connected applications.

**Note:** Registration and access recovery services, the security gateway and the panel do not have to be installed, if you don't intend to use their associated features.

3. DBMS. You can use Couchbase Server, PostgreSQL, Postgres Pro.

**Attention:** Interaction of Blitz Identity Provider with PostgreSQL is performed via JDBC. Any relational DBMS with JDBC support can be used instead of PostgreSQL, but it should be separately agreed with our technical specialists within the framework of the corresponding implementation projects.

- Couchbase Server - recommended for building authentication systems with a peak load of over 1000 requests in a second, more than 1 mln authentications per day and with high fault tolerance requirements.
  - PostgreSQL (or other relational DBMS supporting JDBC) - recommended when creating authentication systems with moderate load and medium requirements for fault tolerance, as well as when using domestic operating systems.
4. Account and password repository. You can use either an existing or specially deployed in your organization repository of accounts for its storage.

Supported:

- LDAP-compliant storage. It can be any server supporting LDAP protocol, as well as Microsoft Active Directory, Samba4, FreeIPA;
- other types of repositories, to connect Blitz Identity Provider to them you need to develop special REST-services.

If you need to deploy a new LDAP directory, it is recommended that you use 389 Directory Server, which is included with the OS, as your LDAP directory.

5. Optional Queue server – used by RabbitMQ. You can also configure the transmission of security events to Kafka. Installing a RabbitMQ queue server is required if the queue server will be used for [transmitting events to adjacent systems](#) (page 254) or as [message broker](#) (page 255).

Deployment is possible in a configuration with [minimal resources](#) (page 10) or in [cluster configuration](#) (page 11).

## 2.1.2 System requirements

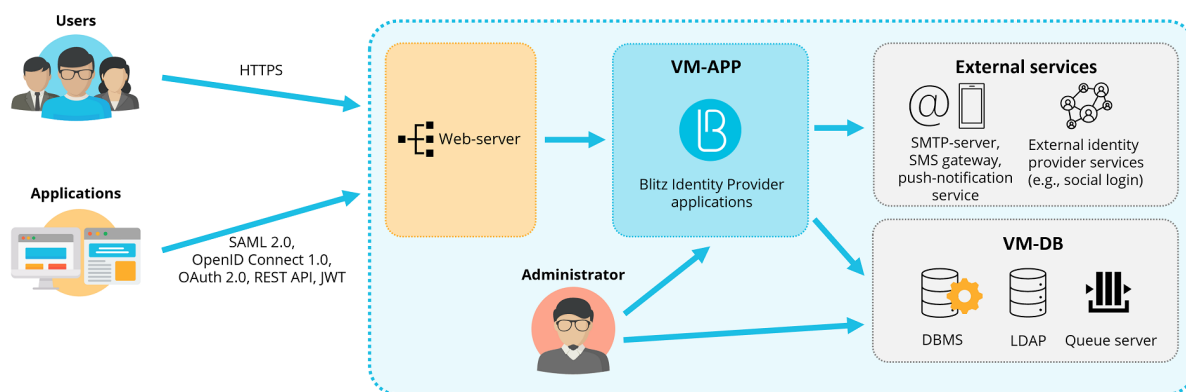
### Operating systems

All Blitz Identity Provider installation options and the server types involved support the following operating systems:

CentOS 7/8 Rocky Linux 8/9 AlmaLinux 8/9 RHEL 7/8/9 Oracle Linux 8/9

### Minimum requirements

Deployments with medium availability and performance requirements are recommended for preparation of test environments and production loops. Follow the scheme below.



2 virtual machines (hereinafter - VMs) with the following characteristics and roles is a minimum requirement for the deployment.

### Minimum sever requirements for deployment

Description	Technical specifications	Software
VM for Applications (VM APP)	4 CPU cores, 8 GB RAM, 50 GB HDD (HDD)	Blitz Identity Provider: <code>blitz-idp</code> , <code>blitz-console</code> , <code>blitz-registration</code> , <code>blitz-recovery</code> , <code>blitz-keeper</code> , <code>blitz-panel</code> ; JDK, <code>nginx</code> , <code>memcached</code>
Database VM (VM DB)	4 CPU cores, 8 GB RAM, 100 GB HDD	PostgreSQL (9.6 or later) or Couchbase Server Community Edition (6.0 or later), 389 Directory Server or FreeIPA; RabbitMQ (optional)

**Required software versions:**

- OpenJDK 11 and Oracle JDK 11;
- Memcached memory manager version 1.4.15 or higher.

**Network connectivity requirements:**

- VM-APP shall be accessible via 80, 443 (HTTP/HTTPS) from user networks;
- VM-APP must have access:
  - to VM-DB via 8091, 8092, 8093, 11209, 11210, 11211, 4369, 21100 to 21199, 11214, 11215, 18091, 18092 (standard Couchbase Server ports), 5432 (standard PostgreSQL port), 389, 636 (standard LDAP ports), 5672 (standard RabbitMQ port);
  - to external identity provider services via 443 port (if used):

**Links to the external identity provider services**

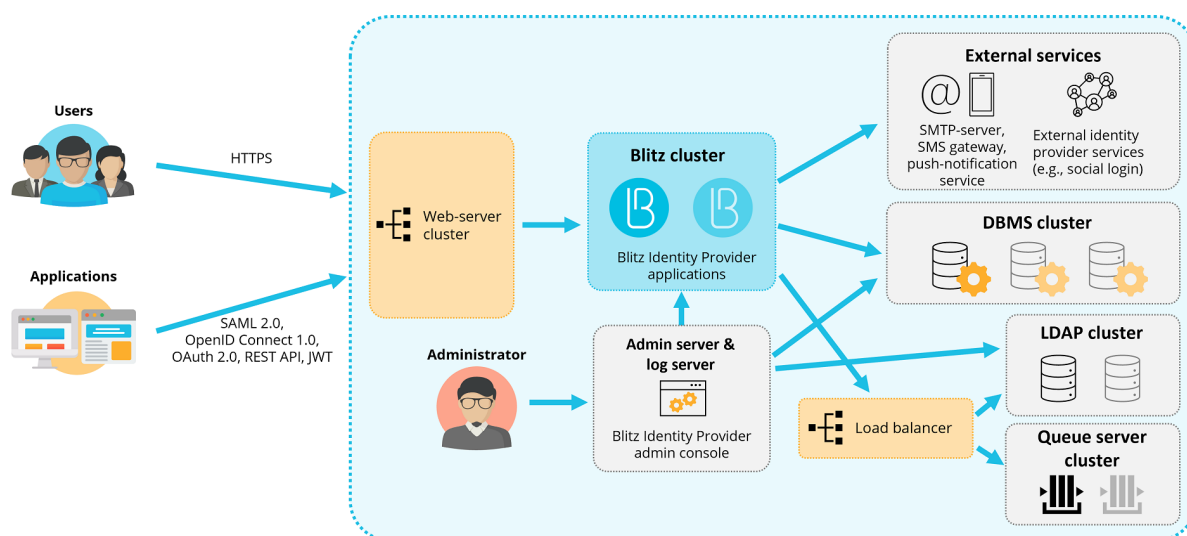
Type	Reference
Social networks	<a href="https://appleid.apple.com">https://appleid.apple.com</a>
	<a href="https://accounts.google.com">https://accounts.google.com</a>
	<a href="https://graph.facebook.com?">https://graph.facebook.com?</a>

- to SMS gateway (if used);
- to SMTP (if used);
- to push notification service (if you use it);
- to the Kafka service (when used to receive security reports).

For VM-APP, you need to create a public DNS name (for example, `auth.domain.ru`) and issue a TLS certificate for `auth.domain.ru` or `*.domain.ru`.

**Recommended requirements for cluster**

Deployment in a cluster configuration is shown in the scheme below. Comply with the given requirements when building productive authentication loops with high availability and peak performance requirements.





For deployment in a cluster configuration, it is recommended to use Virtual Machines (VMs) with the characteristics and functions listed in the table below.

### Recommended server requirements for deployment in a cluster

Description	Q-ty	Technical specifications	Software
VM for web-servers (VM-WEB)	1-2	4 CPU cores, 4 GB RAM, 50 GB HDD	nginx
VM for Blitz Identity Provider applications (VM-APP)	2	4 CPU cores, 8 GB RAM, 50 GB HDD (HDD)	Blitz Identity Provider: blitz-idp, blitz-registration, blitz-recovery, blitz-keeper; blitz-panel; memcached, JDK
VM for console (VM-ADM)	1	2 CPU cores, 4 GB RAM, 100 GB HDD	memcached, JDK; Blitz Identity Provider: blitz-console
VM for DBMS (VM-DB):	2-3	For PostgreSQL: 4 CPU cores, 8 GB RAM, 100 GB HDD (data), 50 GB HDD (system). For Couchbase Server <sup>3</sup> : 8 CPU cores, 16 GB RAM, 500 GB HDD (data), 100 GB SSD (indexes), 50 GB HDD (system).	PostgreSQL software (9.6 or later) or Couchbase Server Community Edition (6.0 or later)
VM for LDAP (VM-LDAP)	2	4 CPU cores, 8 GB RAM, 100 GB HDD	389 Directory Server
VM for Queue server (VM-MQ)	1-2	4 CPU cores, 8 GB RAM, 50 GB HDD (HDD)	RabbitMQ version 3.7.9
VM for the Load balancer (VM-NLB)	1-2	2 CPU cores, 4 GB RAM, 50 GB HDD	HAProxy, keepalived

#### Tip:

- VM-WEB:  
You can use an existing web server to load balance and remove TLS from incoming traffic.
- VM-APP:  
Under heavy load, it is recommended to deploy Blitz Identity Provider services in its own clusters on separate servers.
- VM-ADM:  
It is recommended to configure this server to collect logs from the other servers of the cluster.
- VM-DB:  
For PostgreSQL, it is recommended to allocate one physical server for the main instance and one for standby. For Couchbase Server it is recommended [minimum](https://docs.couchbase.com/server/current/install/deployment-considerations-lt-3nodes.html)<sup>4</sup> 3 VMs.
- VM-LDAP:  
As a storage you can use an existing storage based on LDAP, Microsoft Active Directory, FreeIPA, or any other system for storing accounts and passwords (with the help of a relevant REST connector).

<sup>3</sup> <https://docs.couchbase.com/server/current/install/install-linux.html>

<sup>4</sup> <https://docs.couchbase.com/server/current/install/deployment-considerations-lt-3nodes.html>

- VM-MQ:  
Using a queue server is optional.
- VM-NLB:  
Internal balancer is needed if LDAP and queue server are clustered.

#### Required software versions:

- OpenJDK 11 or Oracle JDK 11;
- Memcached memory manager version 1.4.15 or higher;

#### Network connectivity requirements:

- VM-WEB shall be accessible via 80, 443 (HTTP/HTTPS) from user networks;
- VM-WEB must have access to VM-APP via 9000 (blitz-idp), 9002 (blitz-registration), 9003 (blitz-recovery), 9012 (blitz-keeper), 9013 (blitz-panel) and to VM-ADM via 9001 (blitz-console);
- VM-APP must have access:
  - to other VM-APPS and VM-ADMs via 11211 (memcached);
  - to VM-DB via 8091, 8092, 8093, 11209, 11210, 11211, 4369, 21100 to 21199, 11214, 11215, 18091, 18092 (standard Couchbase Server ports) or 5432 (standard PostgreSQL port);
  - to VM-LDAP (VM-NLB) via 389, 636 (standard LDAP ports);
  - to VM-MQ (VM-NLB) via 5672 (the standard RabbitMQ port);
  - to external identity provider services via 443 port (if used):

#### Links to the external identity provider services

Type	Reference
Social networks	<a href="https://appleid.apple.com">https://appleid.apple.com</a>
	<a href="https://accounts.google.com">https://accounts.google.com</a>
	<a href="https://graph.facebook.com">https://graph.facebook.com</a> <sup>?</sup>

- to the SMS gateway (if used);
- to SMTP (if used);
- to push notification service (if you use it);
- to the Kafka service (when used to receive security reports).
- VM-ADM must have access:
  - to VM-DB via 8091, 8092, 8093, 11209, 11210, 11211, 4369, 21100 to 21199, 11214, 11215, 18091, 18092 (standard Couchbase Server ports) or 5432 (standard PostgreSQL port);
  - to VM-LDAP (VM\_NLB) via 389, 636 (standard LDAP ports);
  - to VM-APP via 22 (ssh), 514 (rsyslog), 873 (rsync), 11211 (memcached);
  - to VM-MQ (VM-NLB) via 5672 (the standard RabbitMQ port);
  - to the Kafka service (when using it to receive security reports)

- from the VM-DB shall have access to other VM-DBs via 8091, 8092, 8093, 11209, 11210, 11211, 4369, 21100 - 21199, 11214, 11215, 18091, 18092 (Couchbase Server ports) or 5432 (PostgreSQL port);
- with VM-LDAP there must be access to other VM-LDAPs via 389, 636 (LDAP ports);
- from the VM-MQ must have access to other VM-MQs via 4369, 35197, 5672.

For VM-APP, you need to create a public DNS name (for example, `auth.domain.ru`) and issue a TLS certificate for `auth.domain.ru` or `*.domain.ru`.

### 2.1.3 General installation instructions

Blitz Identity Provider installation generally proceeds in the order described below.

**Tip:** Depending on the operating system used, there are specifics on how to install the required environment. For convenience, follow [express instructions](#) (page 26).

**Important:** Before getting started with deployment, learn Blitz Identity Provider [deployment architecture](#) (page 9).

#### Step 1. JDK

On the servers designated to install Blitz Identity Provider server software and Blitz Identity Provider admin console, you must install and configure JDK 11 according to the official documentation, using one of the following products:

- OpenJDK 11;

**Note:** To install OpenJDK 11 in CentOS and RHEL, run the command:

```
sudo yum install java-11-openjdk-devel
```

- [Liberica JDK 11](#)<sup>8</sup>;
- [Oracle JDK 11](#)<sup>9</sup>.

#### Step 2. Memcached

**Attention:** The memcached version must be 1.4.15 or higher. The memcached service must be installed on the servers intended for installing Blitz Identity Provider services: `blitz-console`, `blitz-idp`, `blitz-registration`, `blitz-recovery`.

#### CentOS and RHEL

1. Run the command:

```
yum -y install memcached
```

<sup>8</sup> <https://docs.bell-sw.com/liberica-jdk/11.0.23b12/general/install-guide/>

<sup>9</sup> <https://www.oracle.com/java/technologies/javase/jdk11-archive-downloads.html>

2. After installation is complete, add the `memcached` service to the `autorun` and start the service:

```
systemctl enable memcached
systemctl start memcached
```

---

**Important:** The `memcached` service runs on port 11211. Make sure that this port is open on firewalls and can be used to connect between servers with Blitz Identity Provider services.

---

### Step 3. DBMS

#### Couchbase Server Installation

Couchbase Server installation guidelines are provided for CentOS 7 and RHEL 7.

1. You must install Couchbase Server on each of the servers allocated for DBMS installation [according to the instructions<sup>10</sup>](#). The Couchbase Server distribution package is available for [download<sup>11</sup>](#).

---

**Important:** In DEV/TEST environments, it is acceptable to install Couchbase Server on existing servers with Blitz Identity Provider, but in this case you have to take into account that Couchbase Server uses its own built-in Memcached service, and to avoid a conflict you need to adjust the Memcached ports used in Blitz Identity Provider and Couchbase Server.

---

2. Add the Couchbase Server service to the `autorun` and start the service:

```
systemctl enable couchbase-server
systemctl start couchbase-server
```

3. Check if the service is running by executing the command:

```
systemctl status couchbase-server
```

4. Initialize Couchbase Server cluster on each server according to [instructions<sup>12</sup>](#) (the first server initializes the cluster, other servers are included in the cluster). All settings can be set as suggested by default, only you need to set the full server name for each server in `hostname`. It is not recommended to use the IP address of the server as the server name.
5. On any of the hosts in the Couchbase Server cluster, run the script to prepare Couchbase Server to use Blitz Identity Provider. The script is located in the `couchbase` directory in the `resources.zip` archive as part of Blitz Identity Provider distribution kit. Copy the script to any server in the Couchbase Server cluster, go to the directory, and execute the script to create `buckets` that will store Blitz Identity Provider information and indexes for executing Blitz Identity Provider search queries in the database:

```
./cb_init.sh
```

The script will need to be entered during execution:

- Couchbase Server URL name - enter a string like `http://<hostname>:8091`, where `hostname` is the host name of the server from which the script is executed;
- Couchbase Server administrator account login - set during cluster initialization when you perform the previous step of the instructions;

---

<sup>10</sup> <https://docs.couchbase.com/server/current/install/install-linux.html>

<sup>11</sup> <https://www.couchbase.com/downloads>

<sup>12</sup> <https://docs.couchbase.com/server/current/manage/manage-nodes/initialize-node.html>

- Couchbase Server administrator account password - set during cluster initialization when you perform the previous step of the instructions;
- Couchbase Server account login - set during the running of Blitz Identity Provider service connection script;

---

**Tip:** It is recommended to name it `blitz`.

---

- Couchbase Server account password for Blitz Identity Provider application connection.
6. After running the script, make the following settings:
    1. In the Couchbase Server administration console, edit the settings for the number of data copies on different Couchbase instances. To do this, select each bucket in turn in the Buckets menu, click Edit on it and set the Enable setting in the Replicas block and set the number of replicas. For a cluster of 3 servers it is recommended to set 1 for the number of replicas. Then, it is recommended to enable the Enable auto-failover setting in the Settings menu and set the Timeout value to 30 seconds (auto-failover will work only if the DBMS cluster has at least 3 servers and bucket replication is configured).
    2. [Set up a database backup<sup>13</sup>](#).

## PostgreSQL installation and configuration

**Attention:** PostgreSQL must be 9.6 or a later version.

### CentOS and RHEL

PostgreSQL must be installed according to the [instructions<sup>15</sup>](#).

After installing PostgreSQL, run scripts to prepare PostgreSQL to use Blitz Identity Provider. The scripts are located in the `postgres` directory in the `resources.zip` archive as part of Blitz Identity Provider distribution kit. Copy the scripts to the PostgreSQL server, go to the directory, and execute the following commands one by one:

```
su - postgres

createdb blitzdb

psql
CREATE USER blitz WITH ENCRYPTED PASSWORD 'set-your-pwd';
GRANT ALL PRIVILEGES ON DATABASE blitzdb TO blitz;
GRANT ALL ON ALL TABLES IN SCHEMA public TO blitz;

psql -d blitzdb -U blitz -f 000-SCRIPT000.sql
...
psql -d blitzdb -U blitz -f NNN-SCRIPTNNN.sql
```

Instead of `set-your-pwd` you should insert the password that will be used to connect to PostgreSQL.

Instead of `000-SCRIPT000.sql ... NNN-SCRIPTNNN.sql` you should insert the names of scripts from the `postgres/ddl` directory from the `resources.zip` archive. For example:

```
psql -d blitzdb -U blitz -f 000-service-tasks.sql
psql -d blitzdb -U blitz -f 001-init-database.sql
psql -d blitzdb -U blitz -f 002-new_pp_columns.sql
```

(continues on next page)

<sup>13</sup> <https://docs.couchbase.com/server/current/manage/manage-backup-and-restore/manage-backup-and-restore.html>

<sup>15</sup> <https://www.postgresql.org/download/linux/redhat/>

(continued from previous page)

```

psql -d blitzdb -U blitz -f 003-usd_id_table.sql
psql -d blitzdb -U blitz -f 004-usr_auth_table.sql
psql -d blitzdb -U blitz -f 005-usr_agt_table.sql
psql -d blitzdb -U blitz -f 006-usr_htp_hmc_alg.sql
psql -d blitzdb -U blitz -f 007-usr_atr_cfm.sql
psql -d blitzdb -U blitz -f 008-wak.sql
psql -d blitzdb -U blitz -f 009-fix_pp_column.sql
psql -d blitzdb -U blitz -f 010-add_usr_prp.sql
psql -d blitzdb -U blitz -f 011-pp_audit.sql
psql -d blitzdb -U blitz -f 012-geo_to_audit.sql
psql -d blitzdb -U blitz -f 013-tasks.sql
psql -d blitzdb -U blitz -f 014-sec_ch_ua.sql
psql -d blitzdb -U blitz -f 015-5.12.0.sql
psql -d blitzdb -U blitz -f 016-5.13.0.sql
psql -d blitzdb -U blitz -f 017-5.15.0.sql
psql -d blitzdb -U blitz -f 018-5.17.0.sql
psql -d blitzdb -U blitz -f 019-5.18.0.sql
psql -d blitzdb -U blitz -f 020-5.20.0.sql
psql -d blitzdb -U blitz -f 021-5.21.0.sql
psql -d blitzdb -U blitz -f 022-5.23.0.sql

```

After running the script, [set up a database backup](#)<sup>16</sup>.

#### Step 4. RabbitMQ

##### Optional

Installation of the RabbitMQ Queue server is optional and is required if the Queue server is to be used to [pass events to adjacent systems](#) (page 254) or as a [message broker](#) (page 255).

##### CentOS and RHEL

You need to install RabbitMQ according to [instructions](#)<sup>18</sup>.

#### Step 5. Blitz Identity Provider

To install the `blitz-console`, `blitz-idp`, `blitz-registration`, and `blitz-recovery` services, use the unified `blitz-5.X.X.bin` installer.

---

**Important:** You can install the admin console on any server where the Blitz Identity Provider server is installed, but it is recommended that a separate administrative server be dedicated to the installation of the admin console. The [JDK](#) (page 14) and [memcached](#) (page 14) must be installed on the server beforehand.

---

To install `blitz-console`, `blitz-idp`, `blitz-registration`, `blitz-recovery` applications you need to:

1. Copy `blitz-5.X.X.X.bin` file (for example, to ```/tmp` directory) from Blitz Identity Provider distribution kit to the servers intended for installation.
2. Run the `blitz-5.X.X.bin` installer with the following start options:
  - `-i` – list of applications to be installed, separated by a space (for example, `idp console registration recovery`);
  - `-j` – the `JAVA_HOME` value is the directory of JDK installation.

<sup>16</sup> <https://postgrespro.ru/docs/postgresql/9.6/backup-dump#backup-dump-all>

<sup>18</sup> <https://www.rabbitmq.com/install-rpm.html>

It will be installed in directory `/usr/share/identityblitz`.

Listing 1: Installer launch example

```
cd /tmp
chmod +x blitz-5.X.X.bin
./blitz-5.X.X.bin -- -j /opt/oracle/jdk -i "idp console recovery registration"
```

Listing 2: Console during the installation process

```
Verifying archive integrity... 100% MD5 checksums are OK. All good.
Uncompressing Blitz IDP 100%
*****
Application blitz-registration installed
Application blitz-recovery installed
Application blitz-console installed
Application blitz-idp installed
*****
```

3. Create the `blitz_param.txt` file with initial Blitz Identity Provider settings:

#### Couchbase Server

- `DOMAIN` – external domain name where Blitz Identity Provider will be running on;
- `ROOT_CONTEXT` – URL path where Blitz Identity Provider will be running on;

**Note:** If not specified, it will be `/blitz` by default.

- `ADMIN_USER_NAME` – administrator account in Blitz Identity Provider;

**Note:** If not specified, it will be `admin` by default.

- `ADMIN_PASSWORD` – password for the administrator account in Blitz Identity Provider;
- `KEYSTORE_PASSWORD` – password for a key container that will be created during the installation;

**Note:** If the `ADMIN_PASSWORD` and `KEYSTORE_PASSWORD` parameters are not specified, these passwords will automatically be generated and displayed as a result of the configuration script execution.

- `MEMCACHED_SERVERS` – memcached servers addresses;
- `DB_MODE` – DBMS in use: `CB` for Couchbase Server;
- `CB_NODES` – addresses of servers in the Couchbase Server DBMS;
- `CB_USERNAME` – account name in the Couchbase Server DBMS (`blitz` by default);
- `CB_PASSWORD` – account password in the Couchbase Server DBMS;
- `TRUSTED_SERVERS` – addresses of subnets of servers with Blitz Identity Provider services (by default `127.0.0.1/32`).

Listing 3: The example of configuration file

```
DOMAIN=test
MEMCACHED_SERVERS="192.168.122.10 127.0.0.1"
DB_MODE=CB
CB_NODES="192.168.122.20 192.168.122.21 192.168.122.22"
CB_USERNAME=blitz
CB_PASSWORD=12ABcd45
```

## PostgreSQL

- DOMAIN – external domain name where Blitz Identity Provider will be running on;
- ROOT\_CONTEXT – URL path where Blitz Identity Provider will be running on;

**Note:** If not specified, it will be `/blitz` by default.

- ADMIN\_USER\_NAME – administrator account in Blitz Identity Provider;

**Note:** If not specified, it will be `admin` by default.

- ADMIN\_PASSWORD – password for the administrator account in Blitz Identity Provider;
- KEYSTORE\_PASSWORD – password for a key container that will be created during the installation;

**Note:** If the ADMIN\_PASSWORD and KEYSTORE\_PASSWORD parameters are not specified, these passwords will automatically be generated and displayed as a result of the configuration script execution.

- MEMCACHED\_SERVERS – memcached servers addresses;
- DB\_MODE – DBMS in use: PG for PostgreSQL;
- PG\_HOSTNAME – PostgreSQL DBMS address;
- PG\_DB\_NAME – database name in the PostgreSQL DBMS;

---

**Tip:** It is recommended to set `blitzdb`.

---

- PG\_USER\_NAME – account name in the PostgreSQL DBMS;

---

**Tip:** It is recommended to set `blitz`.

---

- PG\_USER\_PASSWORD – account password in the PostgreSQL DBMS;
- TRUSTED\_SERVERS – addresses of subnets of servers with Blitz Identity Provider services (by default `127.0.0.1/32`).

Listing 4: The example of configuration file

```
DOMAIN=test
ROOT_CONTEXT=/blitz
MEMCACHED_SERVERS="127.0.0.1 192.168.122.96"
DB_MODE=PG
```

(continues on next page)



(continued from previous page)

```
PG_HOSTNAME=192.168.122.20
PG_DB_NAME=blitzdb
PG_USERNAME=blitz
PG_PASSWORD=123456
TRUSTED_SERVERS="127.0.0.1/32 192.168.122.96/32 192.168.122.0/24"
ADMIN_USERNAME=admin1
ADMIN_PASSWORD=0123456789
KEYSTORE_PASSWORD=0123456789
```

#### 4. Run Blitz Identity Provider initial setup script with the right path to the `blitz_param.txt` file:

```
/usr/share/identityblitz/blitz-console/bin/configure -f blitz_param.txt
```

The script will prepare the configuration files, generate and display the Blitz Identity Provider administrator login and password, and generate a password for the key container:

```
*****
Your instance is configured on domain: test.loc
The Administration Console available on addresses:
  http://testinstallation.local:9001/blitz/console

Administration user credentials of Console:
  username - admin
  password - 98aAB0D3f2
Your can change user credentials at file - /usr/share/identityblitz/blitz-
↳config/credentials

Create keystore /usr/share/identityblitz/blitz-config/blitz-keystore.bks and
↳generate:
  - JWS (RSA256) keypair - jws_rs256_rsa_default
  - AES (AES128) security key - jdbc

Generated password for keystore: BeEBcd2239
*****
```

**Tip:** If input errors were made when running the installer, so that the installation was performed with incorrect parameters, you can use the following command to delete the files that the installer created so that you can perform a clean installation again:

```
rm -rf /usr/share/identityblitz /etc/default/blitz-* /etc/blitz-* /var/log/
↳identityblitz/ /lib/systemd/system/blitz-*
```

#### 5. Add services to autorun on their respective servers and run them:

```
systemctl enable blitz-console
systemctl start blitz-console
systemctl enable blitz-idp
systemctl start blitz-idp
systemctl enable blitz-registration
systemctl start blitz-registration
systemctl enable blitz-recovery
systemctl start blitz-recovery
```

## Step 6. Configuration files synchronization

Only for installation in a cluster

When you deploy Blitz Identity Provider in a cluster, you must configure synchronization of configuration between Blitz Identity Provider cluster servers:

### Actions to take on the Blitz Identity Provider admin console server

1. Install `rsync` and `incron`:

```
sudo yum install rsync incron
```

2. To switch to user `blitz`:

```
sudo su - blitz
```

3. Generate an ssh-key with the command (it is recommended to choose the default answers for all the prompts by the utility):

```
ssh-keygen
```

4. Read and save the public ssh-key for future use:

```
cat /usr/share/identityblitz/.ssh/id_rsa.pub
```

5. Open the `incrontab` settings:

```
incrontab -e
```

In the opened editor window, insert the following:

```
/usr/share/identityblitz/blitz-config IN_MODIFY,IN_ATTRIB,IN_CREATE,IN_DELETE,
↪IN_CLOSE_WRITE /usr/share/identityblitz/scripts/config_sync.sh ./ $# $%
/usr/share/identityblitz/blitz-config/assets IN_MODIFY,IN_ATTRIB,IN_CREATE,IN_
↪DELETE,IN_CLOSE_WRITE /usr/share/identityblitz/scripts/config_sync.sh assets
↪$# $%
/usr/share/identityblitz/blitz-config/assets/services IN_MODIFY,IN_ATTRIB,IN_
↪CREATE,IN_DELETE,IN_CLOSE_WRITE /usr/share/identityblitz/scripts/config_sync.
↪sh assets $# $%
/usr/share/identityblitz/blitz-config/assets/themes IN_MODIFY,IN_ATTRIB,IN_
↪CREATE,IN_DELETE,IN_CLOSE_WRITE /usr/share/identityblitz/scripts/config_sync.
↪sh assets $# $%
/usr/share/identityblitz/blitz-config/apps IN_MODIFY,IN_ATTRIB,IN_CREATE,IN_
↪DELETE,IN_CLOSE_WRITE /usr/share/identityblitz/scripts/config_sync.sh apps $
↪# $%
/usr/share/identityblitz/blitz-config/saml IN_MODIFY,IN_ATTRIB,IN_CREATE,IN_
↪DELETE,IN_CLOSE_WRITE /usr/share/identityblitz/scripts/config_sync.sh saml $
↪# $%
/usr/share/identityblitz/blitz-config/saml/conf IN_MODIFY,IN_ATTRIB,IN_CREATE,
↪IN_DELETE,IN_CLOSE_WRITE /usr/share/identityblitz/scripts/config_sync.sh.
↪saml $# $%
/usr/share/identityblitz/blitz-config/saml/credentials IN_MODIFY,IN_ATTRIB,IN_
↪CREATE,IN_DELETE,IN_CLOSE_WRITE /usr/share/identityblitz/scripts/config_sync.
↪sh saml $# $%
/usr/share/identityblitz/blitz-config/saml/metadata IN_MODIFY,IN_ATTRIB,IN_
↪CREATE,IN_DELETE,IN_CLOSE_WRITE /usr/share/identityblitz/scripts/config_sync.
↪sh saml $# $%
```

(continues on next page)

(continued from previous page)

```

/usr/share/identityblitz/blitz-config/custom_messages IN_MODIFY,IN_ATTRIB,IN_
↳CREATE,IN_DELETE,IN_CLOSE_WRITE /usr/share/identityblitz/scripts/config_sync.
↳sh custom_messages $# %
/usr/share/identityblitz/blitz-config/custom_messages/dics IN_MODIFY,IN_ATTRIB,
↳IN_CREATE,IN_DELETE,IN_CLOSE_WRITE /usr/share/identityblitz/scripts/config_
↳sync.sh custom_messages $# %
/usr/share/identityblitz/blitz-config/devices IN_MODIFY,IN_ATTRIB,IN_CREATE,IN_
↳DELETE,IN_CLOSE_WRITE /usr/share/identityblitz/scripts/config_sync.sh
↳devices $# %
/usr/share/identityblitz/blitz-config/simple IN_MODIFY,IN_ATTRIB,IN_CREATE,IN_
↳DELETE,IN_CLOSE_WRITE /usr/share/identityblitz/scripts/config_sync.sh simple
↳$# %
/usr/share/identityblitz/blitz-config/certs IN_MODIFY,IN_ATTRIB,IN_CREATE,IN_
↳DELETE,IN_CLOSE_WRITE /usr/share/identityblitz/scripts/config_sync.sh certs $
↳# %
/usr/share/identityblitz/blitz-config/flows/login IN_MODIFY,IN_ATTRIB,IN_
↳CREATE,IN_DELETE,IN_CLOSE_WRITE /usr/share/identityblitz/scripts/config_sync.
↳sh flows $# %
/usr/share/identityblitz/blitz-config/flows/reg IN_MODIFY,IN_ATTRIB,IN_CREATE,
↳IN_DELETE,IN_CLOSE_WRITE /usr/share/identityblitz/scripts/config_sync.sh
↳flows $# %
/usr/share/identityblitz/blitz-config/flows/extIdps IN_MODIFY,IN_ATTRIB,IN_
↳CREATE,IN_DELETE,IN_CLOSE_WRITE /usr/share/identityblitz/scripts/config_sync.
↳sh flows $# %
/usr/share/identityblitz/blitz-config/token_exchange IN_MODIFY,IN_ATTRIB,IN_
↳CREATE,IN_DELETE,IN_CLOSE_WRITE /usr/share/identityblitz/scripts/config_sync.
↳sh token_exchange $# %
/usr/share/identityblitz/blitz-config/token_exchange/rules IN_MODIFY,IN_ATTRIB,
↳IN_CREATE,IN_DELETE,IN_CLOSE_WRITE /usr/share/identityblitz/scripts/config_
↳sync.sh token_exchange $# %

```

6. Create a file `/usr/share/identityblitz/scripts/config_sync.sh` and paste the script into it:

```

#!/bin/bash
app_dir=/usr/share/identityblitz/blitz-config
node_list="NODES_LIST"
for node in $(echo "${node_list}"); do
rsync -r -a --delete ${app_dir}/${1} ${USER}@${node}:${app_dir};
done

```

7. As the value `node_list`, instead of `NODES_LIST`, the list of hostname of the Blitz cluster nodes (except for the Blitz Console node) should be entered. The values should be entered with a space. For example:

```
node_list="app1.local app2.local"
```

8. Make the file `/usr/share/identityblitz/scripts/config_sync.sh` executable:

```
chmod +x /usr/share/identityblitz/scripts/config_sync.sh
```

9. Run `incrontab` by executing the following command as root:

```
systemctl enable incron
systemctl start incron
```

### Actions to take on the other Blitz Identity Provider servers

1. Install `rsync`:

```
sudo yum install rsync
```

2. To switch to user `blitz`:

```
sudo su - blitz
```

3. Run the following script:

```
mkdir .ssh
touch .ssh/authorized_keys
chmod 700 .ssh
chmod 640 .ssh/authorized_keys
```

4. Open the `.ssh/authorized_keys` file with any editor, such as `vim`, and paste the public ssh-key previously obtained from the Blitz Console server.

### Step 7. Web Server

It is recommended to use `nginx` as a web server. A sample configuration file for `nginx` is included in Blitz Identity Provider distribution package - it is the `blitz-idp.conf` file from the `nginx` directory in the `resources.zip` archive. You need to adjust the following configuration blocks, then upload the file to the server with `nginx` (`/etc/nginx/conf.d` directory):

1. Adjust the balancing settings block:

```
upstream blitz-idp {
    server [BLITZ-IDP-NODE-01]:9000 max_fails=3 fail_timeout=120;
    server [BLITZ-IDP-NODE-02]:9000 max_fails=3 fail_timeout=120;
}
upstream blitz-reg {
    server [BLITZ-REG-NODE-01]:9002 max_fails=3 fail_timeout=120;
    server [BLITZ-REG-NODE-02]:9002 max_fails=3 fail_timeout=120;
}
upstream blitz-rec {
    server [BLITZ-REC-NODE-01]:9003 max_fails=3 fail_timeout=120;
    server [BLITZ-REC-NODE-02]:9003 max_fails=3 fail_timeout=120;
}
upstream blitz-console {
    server [BLITZ-CONSOLE-NODE-01]:9001 max_fails=3 fail_timeout=120;
}
```

The parameters have the following designations:

- `[BLITZ-%%-NODE-XX]` - names (hostname) of servers with Blitz Identity Provider services (`blitz-idp`, `blitz-registration`, `blitz-recovery`);
- `[BLITZ-CONSOLE-NODE-01]` is the name (hostname) of the server with Blitz Console.

2. Correct the block of TLS termination settings:

```
ssl_certificate      [BLITZ-SSL-CERT-FILE];
ssl_certificate_key  [BLITZ-SSL-PRIVATEKEY-FILE];
```

The parameters have the following designations:

- `[BLITZ-SSL-CERT-FILE]` - path (full name) to the file with TLS server certificate;

- [BLITZ-IDP-CONSOLE-NODE-01] - path (full name) to the file with TLS-server key.
- Note that Blitz Identity Provider ignores the X-Forwarded-Proto https header if the nginx X-Forwarded-For contains more than one IP address, for example:

```
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
```

In this case, it is recommended to use the following directive value:

```
proxy_set_header X-Forwarded-For $client_ip
```

In this case, `client_ip` is calculated using `map`. The first value from the list will be taken:

```
map $http_x_forwarded_for $client_ip {
    default $remote_addr;
    "~(?<IP>([0-9]{1,3}\.){3}[0-9]{1,3})*" $IP;
    "~(?<IP>([0-9]{1,3}\.){3}[0-9]{1,3}),.*" $IP;
}
```

- Copy the `static_errors` folder with the server error page files to the `/usr/share/nginx/html` folder on the nginx server. The files with examples of error pages can be found in the `|project|` distribution kit - it is the `static_errors` folder in the `resources.zip` archive.

## Step 8. LDAP directory

Optional

See also:

[List of supported directories](#) (page 9).

If you need to deploy a new LDAP directory, it is recommended that you use 389 Directory Server, which is included with the OS, as your LDAP directory:

### CentOS and RHEL

- Execute the installation commands:

```
yum install 389-ds-base 389-adminutil 389-admin 389-admin-console 389-console_
↪389-ds-console
yum install xauth
```

- Set limits according to the 389 Directory Server recommendations:

```
echo "fs.file-max = 64000" >> /etc/sysctl.conf
echo "* soft nofile 8192" >> /etc/security/limits.conf
echo "* hard nofile 8192" >> /etc/security/limits.conf
echo "ulimit -n 8192" >> /etc/profile
```

- Initialize the LDAP directory. Answer the installer's questions:

```
setup-ds-admin.pl
```

- After installation is complete, add the LDAP directory to the autorun and start the service:

```
systemctl enable dirsrv.target
systemctl start dirsrv.target
```

After installing 389 Directory Server, configure it to prepare it for use in conjunction with Blitz Identity Provider. To do this:

1. Copy to the LDAP server the LDAP configuration scripts from Blitz Identity Provider distribution kit (this is the `ldap` folder in the `resources.zip` archive).
2. Execute the initial configuration script `ldap_init.sh` - the script will create the `sub` branch for storing users, service user `reader`, configure user access rights and password policy (perpetual password for service user), create the `blitz-schema` class with attributes `uid`, `mail`, `mobile`, `n`, `name`:

```
chmod +x ldap_init.sh
./ldap_init.sh
```

3. Run the TLS configuration script on the LDAP server (the script creates a copy of the current `NSS DB`, then creates a new `NSS DB`, certificates, and a `pin.txt` file to start the server without entering a password):

```
chmod +x ldap_ssl.sh
./ldap_ssl.sh
```

4. After running the script restart the LDAP directory:

```
systemctl restart dirsrv.target
```

5. If you need to configure and enable global password policies in LDAP, adjust and execute the `ldap_pwdpolicy.sh` script:

```
chmod +x ldap_pwdpolicy.sh
./ldap_pwdpolicy.sh
```

6. If you want to create additional attributes:

1. prepare a text file in which, on each line, write the name of the attribute to be created (i.e. a text file with a column of attributes to be created);
2. run the script to create additional attributes, answer its prompts:

```
chmod +x ldap_add_attr.sh
./ldap_add_attr.sh
```

3. edit the text file at `/etc/dirsrv/slapd-<instance name>/schema/99user.ldif`, add new attributes to objectclass named `blitz-schema` in the `MAY` section;
4. restart the LDAP directory to apply the changes to the directory schema:

```
systemctl restart dirsrv.target
```

### 2.1.4 Express instructions for various operating systems

This section provides express instructions for installing Blitz Identity Provider on various operating systems.

## Limitations when using instructions

**Warning:** The express installation instructions cover a minimal configuration without fault tolerance, placing all components on 1 virtual machine.

**Important:** The operating system must be updated with current patches before work can be performed.

The instructions are given for the case when the virtual machine is connected to the Internet. The instructions use the name `testinstallation.local` as the domain name for installation (it should be corrected). In the scripts used for configuration, the string `CHANGE_ME` is used as the password (it must be corrected). All actions are performed with the privileges of the `root` user.

Blitz Identity Provider distribution kit files must be downloaded and extracted to the `~/tmp/blitz` directory before installation on the server (check the correct version in `BLITZ_REL`):

```
export BLITZ_REL=5.18.0
mkdir -p ~/tmp/blitz
wget -q 'https://nc.idblitz.ru/nextcloud/index.php/s/3W48EBrNXf3R3WC/download?path=
↪%2F'$BLITZ_REL'&files=blitz-'$BLITZ_REL'.bin -O ~/tmp/blitz/blitz-'$BLITZ_REL'.bin
wget -q 'https://nc.idblitz.ru/nextcloud/index.php/s/3W48EBrNXf3R3WC/download?path=
↪%2F'$BLITZ_REL'&files=resources.zip' -O ~/tmp/blitz/resources.zip
unzip ~/tmp/blitz/resources.zip -d ~/tmp/blitz
find ~/tmp/blitz -name *.sh -o -name *.bin|xargs chmod +x
```

## Rocky Linux, AlmaLinux, Oracle Linux, RHEL

**Important:** See [limitations of](#) (page 26) when using express instructions.

The list of operating systems for which the instructions for installation and their designation in this section are given:

- Rocky 8: Rocky Linux 8;
- Alma 8: AlmaLinux 8;
- Oracle 8: Oracle Linux 8;
- RHEL 8: RHEL 8;
- Rocky 9: Rocky Linux 9;
- Alma 9: AlmaLinux 9;
- Oracle 9: Oracle Linux 9;
- RHEL 9: RHEL 9.

### Step 1. JDK

#### Rocky, Alma, Oracle, RHEL 8

Install the distribution kit:

```
dnf install java-11-openjdk-devel
```

#### Rocky, Alma, Oracle, RHEL 9

Install the distribution kit:

```
dnf install java-11-openjdk-devel
```

### Step 2. Memcached

#### Rocky, Alma, Oracle, RHEL 8

Install the distribution kit:

```
dnf install memcached
```

Start the service:

```
systemctl enable memcached && systemctl start memcached
```

#### Rocky, Alma, Oracle, RHEL 9

Install the distribution kit:

```
dnf install memcached
```

Start the service:

```
systemctl enable memcached && systemctl start memcached
```

### Step 3. PostgreSQL

#### Rocky, Alma, Oracle, RHEL 8

Install the distribution kit:

```
dnf install postgresql
```

Initialize the DBMS with the command:

```
postgresql-setup initdb
```

Add permission in `/var/lib/pgsql/data/pg_hba.conf` for the `blitz` user to connect to the database:



```
host blitzdb blitz 127.0.0.1/32 scram-sha-256
```

Specify the password encryption algorithm in `/var/lib/pgsql/data/postgresql.conf`:

```
password_encryption = scram-sha-256
```

Start the service:

```
systemctl enable postgresql && systemctl start postgresql
```

Connect to the DBMS and perform initial configuration

```
su - postgres
psql

create database blitzdb;
create user blitz with encrypted password 'CHANGE_ME';
grant ALL PRIVILEGES ON DATABASE blitzdb to blitz;
grant ALL on ALL tables in schema public to blitz;
```

Return to the root user shell and execute the scripts for creating and updating the blitzdb database structure:

```
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/000-service-tasks.
↪sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/001-init-database.
↪sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/002-new_pp_columns.
↪sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/003-usd_id_table.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/004-usr_auth_table.
↪sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/005-usr_agt_table.
↪sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/006-usr_htp_hmc_alg.
↪sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/007-usr_atr_cfm.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/008-wak.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/009-fix_pp_column.
↪sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/010-add_usr_prp.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/011-pp_audit.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/012-geo_to_audit.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/013-tasks.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/014-sec_ch_ua.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/015-5.12.0.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/016-5.13.0.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/017-5.15.0.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/018-5.17.0.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/019-5.18.0.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/020-5.20.0.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/021-5.21.0.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/022-5.23.0.sql
```

**Rocky, Alma, Oracle, RHEL 9**

Install the distribution kit:

```
dnf install postgresql-server
```

Initialize the DBMS with the command:

```
postgresql-setup -initdb -unit postgresql
```

Add permission in `/var/lib/pgsql/data/pg_hba.conf` for the `blitz` user to connect to the database:

```
host blitzdb blitz 127.0.0.1/32 scram-sha-256
```

Specify the password encryption algorithm in `/var/lib/pgsql/data/postgresql.conf`:

```
password_encryption = scram-sha-256
```

Start the service:

```
systemctl enable postgresql && systemctl start postgresql
```

Return to the `root` user shell and execute the scripts for creating and updating the `blitzdb` database structure:

```
su - postgres
psql

create database blitzdb;
create user blitz with encrypted password 'CHANGE_ME';
grant ALL PRIVILEGES ON DATABASE blitzdb to blitz;
grant ALL on ALL tables in schema public to blitz;
```

Execute the scripts for creating and updating the `blitzdb` database structure:

```
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/000-service-tasks.
↵sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/001-init-database.
↵sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/002-new_pp_columns.
↵sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/003-usd_id_table.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/004-usr_auth_table.
↵sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/005-usr_agt_table.
↵sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/006-usr_htp_hmc_alg.
↵sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/007-usr_atr_cfm.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/008-wak.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/009-fix_pp_column.
↵sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/010-add_usr_prp.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/011-pp_audit.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/012-geo_to_audit.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/013-tasks.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/014-sec_ch_uu.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/015-5.12.0.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/016-5.13.0.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/017-5.15.0.sql
```

(continues on next page)

(continued from previous page)

```
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/018-5.17.0.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/019-5.18.0.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/020-5.20.0.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/021-5.21.0.sql
psql -U blitz -h 127.0.0.1 blitzdb -f ~/tmp/blitz/postgres/ddl/022-5.23.0.sql
```

## Step 4. RabbitMQ

### Rocky, Alma, Oracle, RHEL 8

Prepare a configuration file with repositories for RabbitMQ in `/etc/yum.repos.d/rabbitmq.repo`:

```
##
## Zero dependency Erlang
##

[rabbitmq_erlang]
name=rabbitmq_erlang
baseurl=https://packagecloud.io/rabbitmq/erlang/el/8/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
# PackageCloud's repository key and RabbitMQ package signing key
gpgkey=https://packagecloud.io/rabbitmq/erlang/gpgkey

https://github.com/rabbitmq/signingkeys/releases/download/2.0/rabbitmq-release-
↪signing-key.asc
sslverify=1
sslcert=/etc/pki/tls/certs/ca-bundle.crt
metadata_expire=300

##
## RabbitMQ server
##

[rabbitmq_server]
name=rabbitmq_server
baseurl=https://packagecloud.io/rabbitmq/rabbitmqserver/el/8/$basearch
repo_gpgcheck=1
gpgcheck=0
enabled=1
# PackageCloud's repository key and RabbitMQ package signing key
gpgkey=https://packagecloud.io/rabbitmq/rabbitmq-server/gpgkey

https://github.com/rabbitmq/signingkeys/releases/download/2.0/rabbitmq-release-
↪signing-key.asc
sslverify=1
sslcert=/etc/pki/tls/certs/ca-bundle.crt
metadata_expire=300
```

Install the distribution kit:

```
dnf install rabbitmq-server
```

Start the service:

```
systemctl enable rabbitmq-server && systemctl start rabbitmq-server
```

Prepare a queue for interaction:

```
rabbitmqctl add_user blitz CHANGE_ME
rabbitmqctl set_permissions blitz ".*" ".*" ".*"
rabbitmq-plugins enable rabbitmq_management
curl -vvk 127.0.0.1:15672/cli/rabbitmqadmin >rabbitmqadmin
chmod +x rabbitmqadmin
./rabbitmqadmin declare exchange name=blitz-tasks-exh type=direct
./rabbitmqadmin declare queue name=blitz-tasks durable=true
./rabbitmqadmin declare binding source="blitz-tasks-exh"
destination_type="queue" destination="blitz-tasks"
routing_key="blitz-tasks"
```

### Rocky, Alma, Oracle, RHEL 9

Prepare a configuration file with repositories for RabbitMQ in `/etc/yum.repos.d/rabbitmq.repo`:

```
##
## Zero dependency Erlang
##

[rabbitmq_erlang]
name=rabbitmq_erlang
baseurl=https://packagecloud.io/rabbitmq/erlang/el/9/$basearch
repo_gpgcheck=1
gpgcheck=1
enabled=1
# PackageCloud's repository key and RabbitMQ package signing key
gpgkey=https://packagecloud.io/rabbitmq/erlang/gpgkey

https://github.com/rabbitmq/signingkeys/releases/download/2.0/rabbitmq-release-
↪signing-key.asc
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
metadata_expire=300

##
## RabbitMQ server
##

[rabbitmq_server]
name=rabbitmq_server
baseurl=https://packagecloud.io/rabbitmq/rabbitmqserver/el/9/$basearch
repo_gpgcheck=1
gpgcheck=0
enabled=1
# PackageCloud's repository key and RabbitMQ package signing key
gpgkey=https://packagecloud.io/rabbitmq/rabbitmq-server/gpgkey

https://github.com/rabbitmq/signingkeys/releases/download/2.0/rabbitmq-release-
↪signing-key.asc
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
metadata_expire=300
```

Install the distribution kit:

```
dnf install rabbitmq-server
```

Start the service:

```
systemctl enable rabbitmq-server && systemctl start rabbitmq-server
```

Prepare a queue for interaction:

```
rabbitmqctl add_user blitz CHANGE_ME
rabbitmqctl set_permissions blitz ".*" ".*" ".*"
rabbitmq-plugins enable rabbitmq_management
curl -vvk 127.0.0.1:15672/cli/rabbitmqadmin >rabbitmqadmin
chmod +x rabbitmqadmin
./rabbitmqadmin declare exchange name=blitz-tasks-exh type=direct
./rabbitmqadmin declare queue name=blitz-tasks durable=true
./rabbitmqadmin declare binding source="blitz-tasks-exh"
destination_type="queue" destination="blitz-tasks"
routing_key="blitz-tasks"
```

## Step 5. 389 Directory Server

### Rocky, Alma, Oracle, RHEL 8

Install the distribution kit:

```
dnf module enable 389-directory-server:stable
dnf install 389-ds-base
```

Enable automatic startup of the service:

```
systemctl enable dirsrv.target
```

Initialize the LDAP directory:

```
dscreate interactive
```

Perform the initial directory configuration:

```
/tmp/blitz/ldap/ldap_init.sh
```

### Rocky, Alma, Oracle, RHEL 9

Install the distribution kit:

```
dnf install 389-ds-base
```

Enable automatic startup of the service:

```
systemctl enable dirsrv.target
```

Initialize the LDAP directory:

```
dscreate interactive
```

Perform the initial directory configuration:

```
/tmp/blitz/ldap/ldap_init.sh
```

## Step 6. Nginx

### Rocky, Alma, Oracle, RHEL 8

Install the distribution kit:

```
dnf install nginx
```

Copy the files for use:

```
cp /tmp/blitz/nginx/blitz-idp.conf /etc/nginx/conf.d/  
cp -R /tmp/blitz/static_errors /usr/share/nginx/html
```

Enable automatic startup of the service:

```
systemctl enable nginx
```

### Rocky, Alma, Oracle, RHEL 9

Install the distribution kit:

```
dnf install nginx
```

Copy the files for use:

```
cp /tmp/blitz/nginx/blitz-idp.conf /etc/nginx/conf.d/  
cp -R /tmp/blitz/static_errors /usr/share/nginx/html
```

Enable automatic startup of the service:

```
systemctl enable nginx
```

## Step 7. Blitz Identity Provider

### Rocky, Alma, Oracle, RHEL 8

Install the distribution kit (specify the correct version in the file name, the correct `JAVA_HOME` and the set of applications to install):

```
/tmp/blitz/blitz-5.X.X.bin -- -j <JAVA_HOME> -i "idp console recovery registration"
```

Create the `blitz_param.txt` configuration file with the following content and modify it according to your settings:

```
DOMAIN=testinstallation.local  
MEMCACHED_SERVERS="127.0.0.1"  
DB_MODE=PG  
PG_HOSTNAME=127.0.0.1  
PG_DB_NAME=blitzdb  
PG_USERNAME=blitz  
PG_PASSWORD=12ABcd45
```

Run Blitz Identity Provider initial setup script with the right path to the `blitz_param.txt` file:

```
/usr/share/identityblitz/blitz-console/bin/configure -f blitz_param.txt
```

The script will prepare the configuration files, generate and display the Blitz Identity Provider administrator login and password, and generate a password for the key container:

```
*****
Your instance is configured on domain: test.loc
The Administration Console available on addresses:
  http://testinstallation.local:9001/blitz/console

Administration user credentials of Console:
  username - admin
  password - 98aAB0D3f2
Your can change user credentials at file - /usr/share/identityblitz/blitz-config/
↳credentials

Create keystore /usr/share/identityblitz/blitz-config/blitz-keystore.bks and
↳generate:
  - JWS (RSA256) keypair - jws_rs256_rsa_default
  - AES (AES128) security key - jdbc

Generated password for keystore: BeEBcd2239
*****
```

In case of using keys created during the installation phase, restart nginx:

```
systemctl restart nginx
```

Add a mapping between the loopback interface address and the domain name specified at installation in `/etc/hosts`:

```
127.0.0.1 localhost.localdomain localhost testinstallation.local
```

Start the services:

```
systemctl enable blitz-idp && systemctl start blitz-idp
systemctl enable blitz-console && systemctl start blitz-console
systemctl enable blitz-registration && systemctl start blitz-registration
systemctl enable blitz-recovery && systemctl start blitz-recovery
```

After successfully completing the installation and configuration of Blitz Identity Provider, it is possible to connect to the admin console using the domain name specified during the installation phase of the distribution kit, for example, `https://testinstallation.local/blitz/console`.

### Rocky, Alma, Oracle, RHEL 9

Install the distribution kit (specify the correct version in the file name, the correct `JAVA_HOME` and the set of applications to install):

```
/tmp/blitz/blitz-5.X.X.bin -- -j <JAVA_HOME> -i "idp console recovery registration"
```

Create the `blitz_param.txt` configuration file with the following content and modify it according to your settings:

```
DOMAIN=testinstallation.local
MEMCACHED_SERVERS="127.0.0.1"
DB_MODE=PG
```

(continues on next page)

(continued from previous page)

```
PG_HOSTNAME=127.0.0.1
PG_DB_NAME=blitzdb
PG_USERNAME=blitz
PG_PASSWORD=12ABcd45
```

Run Blitz Identity Provider initial setup script with the right path to the `blitz_param.txt` file:

```
/usr/share/identityblitz/blitz-console/bin/configure -f blitz_param.txt
```

The script will prepare the configuration files, generate and display the Blitz Identity Provider administrator login and password, and generate a password for the key container:

```
*****
Your instance is configured on domain: test.loc
The Administration Console available on addresses:
  http://testinstallation.local:9001/blitz/console

Administration user credentials of Console:
  username - admin
  password - 98aAB0D3f2
You can change user credentials at file - /usr/share/identityblitz/blitz-config/
↳credentials

Create keystore /usr/share/identityblitz/blitz-config/blitz-keystore.bks and
↳generate:
  - JWS(RSA256) keypair - jws_rs256_rsa_default
  - AES(AES128) security key - jdbc

Generated password for keystore: BeEBcd2239
*****
```

In case of using keys created during the installation phase, restart nginx:

```
systemctl restart nginx
```

Add a mapping between the loopback interface address and the domain name specified at installation in `/etc/hosts`:

```
127.0.0.1 localhost.localdomain localhost testinstallation.local
```

Start the services:

```
systemctl enable blitz-idp && systemctl start blitz-idp
systemctl enable blitz-console && systemctl start blitz-console
systemctl enable blitz-registration && systemctl start blitz-registration
systemctl enable blitz-recovery && systemctl start blitz-recovery
```

After successfully completing the installation and configuration of Blitz Identity Provider, it is possible to connect to the admin console using the domain name specified during the installation phase of the distribution kit, for example, `https://testinstallation.local/blitz/console`.



## 2.1.5 The first steps after installation

The section contains information that you may need after the Blitz Identity Provider installation.

### Configure launch options for Blitz Identity Provider services

The following Java options are available for Blitz Identity Provider applications to define enabling special modes of application operation and override the default modes of operation:

**Attention:** It is recommended that you consult with Blitz Identity Provider technical support before installing options.

- `blitz.login.cookie.sameSite` - specifies the flag with which session cookies should be created in Blitz Identity Provider. By default, cookies are created with the flag `sameSite=Lax`. This can be overridden to `None`.
- `blitz.login.outside.flow.callback.ttl.sec` - specifies the time to wait for a response from an external authentication method called from Blitz Identity Provider. The default value is 300 seconds.
- `blitz.login.mus.cookie.unused.ttl.sec` - sets the lifetime of the cookie responsible for memorizing the list of logged in users in the current browser. The default value corresponds to 365 days (the value is set in seconds);
- `blitz.login.bua.cookie.ttl.sec` - sets the validity time of the cookie used to remember the user's browser. The default value corresponds to 365 days (the value is set in seconds);
- `blitz.login.setLastAuth.disabled` - allows to disable writing the time of the last user authentication to the database. By default, the time of the last user authentication is written to the database. Disabling recording of the last authentication time allows to increase database performance, but does not allow to use the [function of blocking accounts by inactivity](#) (page 264);
- `blitzDispatchedQueues` - specifies the name of the queue from which Blitz Identity Provider processes tasks for sending emails, user registration and password recovery. The default queue name is `default`;
- `blitz.stores.united.u-cache.ttlInSec` - the expiration time of the account data cache provided via the REST API. The default is 1 second;
- `blitz.csrf.cookie.ttlInSec` - specifies the validity time of the cookie preventing CSRF. The default corresponds to 6 hours (the value is set in seconds). This is the maximum time from the moment the user opens the page until the completed page is executed by the user to the server;
- `blitz.jdbc.cols.types.strings` - specifies the column type used to store string attributes in the relational DBMS (PostgreSQL). The `text` type is used by default;
- `blitz.jdbc.pool.stat-period` - specifies the frequency at which JDBC usage statistics are logged. The default is 300 seconds;
- `saml.numThreads` - specifies the number of threads that Blitz Identity Provider processes SAML input requests. The default is 32 threads;
- `blitz.oauth.exchange.rules.fs.cache.capacity` - specifies the cache size used by Blitz Identity Provider to check microservice access rules. The default cache size is 10000 checks;
- `blitz.oauth.dyn.reg.clientSecretLength` - specifies the size of `client_secret` generated when dynamically registering a pair of `client_id` and `client_secret`. By default, `client_secret` is generated with a size of 15 characters.
- `blitz.oauth.dyn.reg.clientAttachingTtlInSec` - specifies the time during which the `client_id` and `client_secret` pair generated during dynamic registration should be associated with the user (if the pair is not associated with the user during this time, it will be canceled). The default corresponds to 1 hour (the value is specified in seconds).

- `blitz.session.checkRemoteAddress.disabled` - set `true` to disable checking the equality of the session and the incoming request IP addresses (recommended if you have users with dynamic IP addresses).
- `blitz.webauthn.residentKey.preferred` - if the option is specified, security keys are registered with the parameter `residentKey=preferred`. In this case, if the option is set as `true`, then `requireResidentKey=true`, and if the option is `false`, then `requireResidentKey=false`.
- `blitz.ldap.store.extension.class` - passing `com.identityblitz.idp.store.ldap.custom.PasswordMigrationExt` to the option enables the password migration mode.
- `blitz.ldap.store.extension.PasswordMigrationExt.passwordHashAttr` - specifies the name of the LDAP attribute that stores the password hash for the password migration option. The hash must contain the `{bcrypt}` prefix for password migration from hashes with `bcrypt` algorithm.
- `extensionsDir` is the address of the directory with [extension modules](#) (page 247).
- `metrics` – allows you to disable gathering performance metrics in the **Prometheus** format. To do so, set the value to `false`. By default, metric gathering is enabled
- `couchbase.durability.mode` - specifies the mode of data saving in Couchbase Server. In case of using Couchbase Server version 6.0.1 or older, `clientVerified` mode must be used. If you are using Couchbase Server versions 6.5, 7.0 or newer, `clientVerified` mode cannot be used. The parameter in Couchbase Server versions 6.5, 7.0 becomes optional (in the absence of the parameter, `majority` mode is used) and allows you to select the required data retention assurance mode in a cluster with replication from the following [options](#)<sup>19</sup>:
  - `disabled` - waiting for memory-only writes on the primary node of the cluster;
  - `majority` - waiting for memory writes on the primary node and most replicas;
  - `majorityAndPersistActive` - waiting to write to disk on the primary node and write to memory for most replicas;
  - `persistToMajority` - waiting to write to disk on the primary node and in most replicas.
- `akka.http.parsing.max-uri-length` - sets the maximum length of URL in the browser string. In some cases it may be necessary to increase the string size, then it is recommended to set `16k` in this parameter.
- `akka.http.parsing.max-header-value-length` - sets the maximum allowed HTTP header size. If it's necessary to increase the header size, set `16k` in this parameter.
- `akka.coordinated-shutdown.phases.service-stop.timeout` - sets the waiting time after receiving the command to stop the service, during which the service can complete the tasks taken into work. If you use the message broker built into Blitz Identity Provider, it is recommended to set the parameter to `30s` for the service.
- `memcached.locator.tries` - defines the number of attempts to find a working Memcached server if the system is processing an access failure to Memcached server.

**Warning:** It is not guaranteed that the options used will be preserved in future versions of Blitz Identity Provider.

To set options with values different from the default values, it is necessary to edit the `/etc/default/blitz-idp` file. Set the necessary `JAVA_OPTS` in it. Below is an example of a file in which the `blitz.csrf.cookie.ttlInSec` and `blitz.login.cookie.sameSite` options are also set among the Java options. After changing `JAVA_OPTS`, you must restart the Blitz Identity Provider services on which the changes are made.

<sup>19</sup> <https://docs.couchbase.com/server/current/learn/data/durability.html>

```

export JAVA_HOME=/usr/java/default
export PIDFILE=/usr/share/identityblitz/blitz-idp/RUNNING_PID
export JAVA_OPTS="-server -Xms512m -Xmx1G -XX:MaxMetaspaceSize=512m -Xmn256m -Dcom.
↪couchbase.connectTimeout=30000 -Dakka.http.parsing.max-uri-length=16k"
export JAVA_OPTS="$JAVA_OPTS -Dblitz.csrf.cookie.ttlInSec=36000 -Dblitz.login.
↪cookie.sameSite=None -Dplay.filters.headers.frameOptions=null"

```

### Logging in to Admin console

After installing Blitz Identity Provider, the basic system configuration is performed in the Admin Console, which can be accessed from the link indicated in the product installation results. For the first login to the Admin console, you must use the login and password generated at the time of installation of the Admin Console.

Usually the link is of the form:

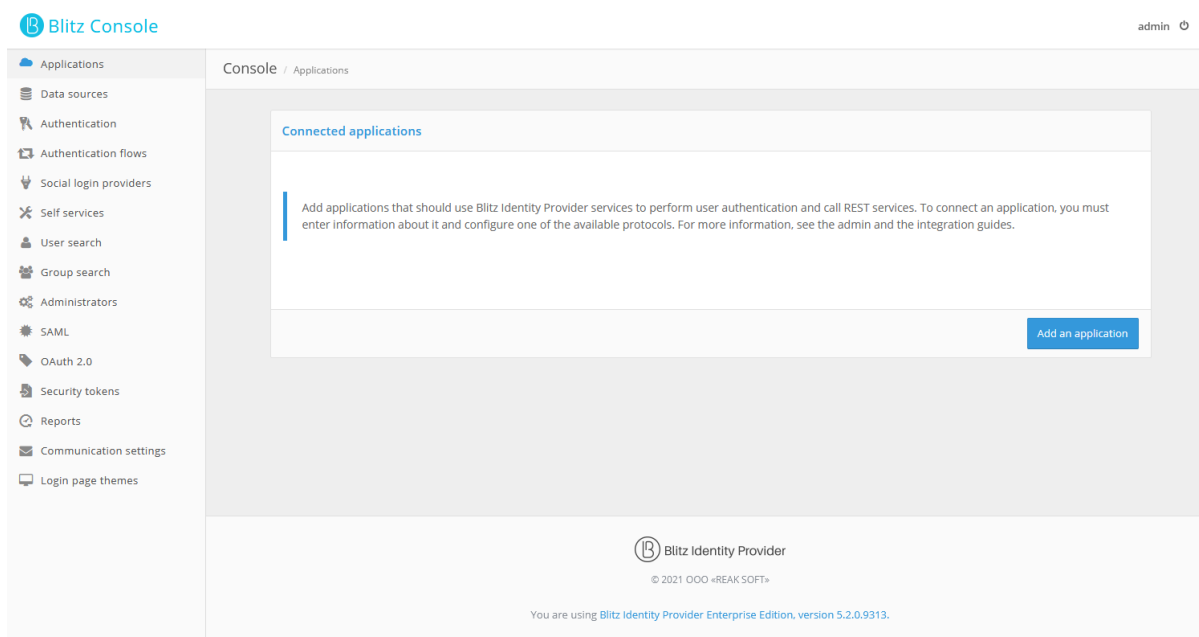
```
https://<blitz_domain>/blitz/console
```

or

```
http://<blitz_console_host>:9001/blitz/console
```

The standard view of the Admin console login screen is shown in the figure:

After successful login, the main page of the Admin Console opens, as shown below. You can navigate between the various Blitz Identity Provider settings using the menu on the left side of the screen.

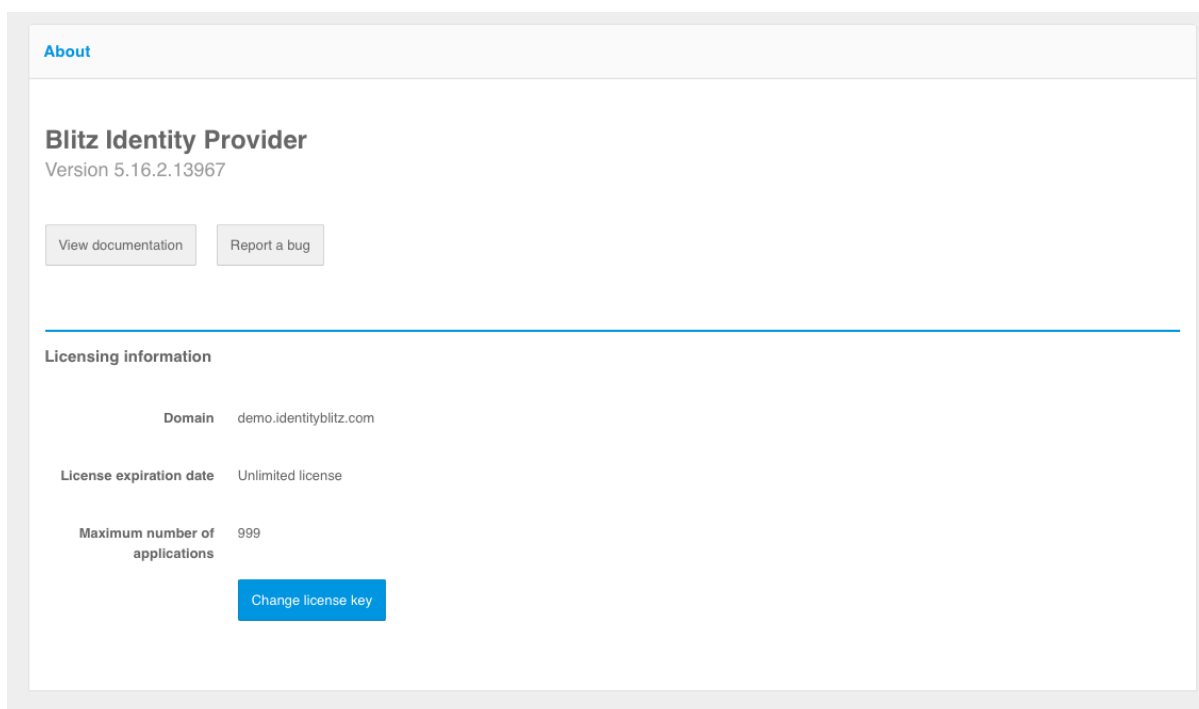


## License key installation

If you click on the [You are using ..., version ...](#) link in the footer of any Blitz Identity Provider admin console page, the screen below will be displayed.

On this screen, you can view the version number of your current Blitz Identity Provider, go to the software documentation site, and the feedback form.

In the License information block you can see the license expiration date and the maximum number of applications allowed by the license. If you click the [Change license](#) button, you can enter a new license key.



After you have installed the new license key it is recommended that you restart Blitz Identity Provider applications.

You can also set the license key by editing the `blitz.conf` configuration file in the `/usr/share/identityblitz/blitz-config` directory. You need to find the `blitz.prod.local.idp.license` configuration block and adjust it as follows (set the license key in the `key` parameter):

```
"license" : {
  "key" : "MEQC...U"
}
```

### Administrator account management

After installing Blitz Identity Provider, it is recommended that you create additional administrator accounts, assign passwords and administrative roles to them. You can manage administrator accounts in the Administrators section.

The screenshot shows the 'Administrators' management page. It features a table with the following structure:

Login	Roles	Password
admin	root	Change password

Below the table, there is a link '+ Create an administrator account' and a 'Save' button at the bottom right.

The following actions are available under Administrators section:

- creation and deletion of administrator accounts;
- change of the administrator account passwords;
- assignment and revoke of administrator functions.

By default, the roles listed in the table are available in Blitz Identity Provider. You can reconfigure existing roles or create new roles through the settings in the `credentials` configuration file.

### Standard administrator roles in Blitz Identity Provider

Role	Available sections of the Admin console
superuser ( <code>root</code> )	Everything is accessible
IS administrator ( <code>security</code> )	Administrators, Events
system administrator ( <code>sysadmin</code> )	Data sources, Authentication, Authentication flows, Identity providers, SAML, OAuth 2.0, Devices, Messages
application administrator ( <code>app_admin</code> )	Applications
Interface administrator ( <code>ui_admin</code> )	Self-services, Login page themes
TA administrator ( <code>support</code> )	Users, Groups, Access rights, Events

In addition to the standard identification and authentication of administrators by login and password when logging into the admin console, it is possible to configure the use of identification and authentication of users to the admin console using the Blitz Identity Provider authentication server. The settings are made through the `console.conf` configuration file.

## Restarting Blitz Identity Provider services

To restart the Blitz Identity Provider services, use the following command:

```
systemctl restart APP_NAME
```

Instead of `APP_NAME`, you must specify the name of the application to be restarted: `blitz-console`, `blitz-idp`, `blitz-registration`, `blitz-recovery`, `blitz-keeper`.

Example of a command to restart an authentication service application:

```
systemctl restart blitz-idp
```

## Deleting files used for installation

When you launch it for the first time, Blitz Identity Provider encrypts the administrator passwords and DBMS connection passwords created during installation. At the same time, the initial configuration file is copied to the `/usr/share/identityblitz/blitz-config/.snapshot` directory. It is recommended to delete the `blitz_param.txt` files used during the installation process and the `blitz.conf` copies. To do so, run the command:

```
rm blitz_param.txt /usr/share/identityblitz/blitz-config/.snapshot/blitz.conf.*
```

## 2.2 Basic configuration

### 2.2.1 User account attributes

A user account in Blitz Identity Provider is described by a set of attributes. This section is dedicated to all aspects of their management.

#### What is an account attribute?

A user account is defined by a set of attributes.

The attribute values are set in the following ways:

- are read from *connected attribute stores* (page 47);
- are read from the Blitz Identity Provider database;

**Note:** The attribute is read and saved in the database if no attribute mapping is configured for the attribute in the connected attribute store.

- computed from other attributes or filled with constant values.

**Tip:** You can compute the attribute `user domain` from an email address `email`, or create a composite attribute `full name` from the individual attributes with the surname, first name and middle name of the user.

The configuration of attributes consists of:

- configuring stored attributes, i.e., those maintained in connected repositories or in the Blitz Identity Provider database;

- configuring computable attributes, i.e., those that must take a constant value or that are computed by rules.
- configuring input value conversion rules that allow you to convert attribute values when they are changed (e.g., when they are edited by the user or during invoking of the corresponding API);
- configuring output value conversion rules that allow to perform additional transformations with the computed attributes;
- configuring attribute assignment - definition of the identifier in the system and attributes, responsible for mobile phone number, e-mail address.

**Attention:** For Blitz Identity Provider to work correctly, as a minimum, the following configurations must be performed:

- necessary attributes are configured;
- one of the attributes is defined as an identifier.

## Configuring the available attributes

### Stored attributes

You must go to the Data Sources block in the Stored attributes section and take the following steps:

- add a new attribute by clicking the `+Add attribute` link;
- specify the attribute name to be used in Blitz Identity Provider; The name of the attribute may be different from its name in the external repository - in latter case, you must specify the conversion rule in the [settings](#) (page 47) of this repository;
- specify the value type - data type format (`String`, `Number`, `Boolean`, `Bytes`, `Array of Strings`);
- set the attribute's parameters:
  - whether it is possible to search for it (the `Srch` column);

---

**Tip:** If it is an attribute from a connected repository, it is recommended to create a search index for it.

---

- whether the attribute is mandatory (the `Mand` column);
- whether the attribute's value should be unique in the system (column `Uniq`).

After adding an attribute, it is not allowed to change its name. If it is required to rename an attribute, delete, and create a new one.

---

**Important:** In the Users section of the [user card](#) (page 139), the attributes will be shown in the order in which they were created. It is not possible to change the order of attributes via the admin console. If you need to change the order of attributes, you must manually reorder them in the `blitz.conf` configuration file in the `blitz.prod.local.idp.id-attrs` settings section. In order to display their text names in the Users section instead of system attribute names, taking into account the user interface language, it is necessary to [define](#) (page 234) for the created attributes in `messages` lines with the description of attribute names for the used languages. The strings must have the form `custom.user.attr.name.<attribute name>`.

---

When you create a new attribute, a mapping of the new attribute is also automatically created in all connected attribute storages to an attribute with the same name. After creating new attributes, you have to check and edit the mapping configurations in the connected storages. If the attribute is not expected to be read from the storage, you have to delete the mapping line - in this case, the attribute will be stored in Blitz Identity Provider database.

**Important:** If PostgreSQL is used as the DBMS, create a column in the `USR_ATTR` table as well as in the `USR` table (if [internal storage](#) (page 46) is used). The column name must correspond to the name of the attribute to be added, but be normalized from `lowerCamelCase` into `UPPERCASE_SEPARATED_BY_UNDERSCORE`, e.g., `middleName` -> `MIDDLE_NAME`. The column type must be chosen based on the type of the attribute value:

- column with `text` type for attributes with `String` and `Bytes` type (in this case the value will be stored in Base64);
- column with `text[]` type for attribute with `Array of strings` type;
- a column with a suitable numeric type (`bigint`, `integer`, `smallint`) for attributes with `Number` type;
- a column with `bool` type for an attribute with `Boolean` type.

#### Stored attributes

Define the user account attributes. To do it, specify the *name* - the unique name of the attribute in the system. The name of the attribute may be different from its name in the external storage. In the latter case specify the conversion rule in the settings of this storage.

Also specify the *value type* - the data type of the attribute.

Specify which attributes are:

- *searchable (Search)* - these attributes will be taken into account when searching for an account in the "Users" section. When using external storage for these attributes you should provide an index
- *mandatory (Mand.)* - these attributes must be specified when registering the user and can not be deleted later
- *unique (Uniq.)* - the values of these attributes must be unique in the system.

Attribute name	Format	Srch	Mand.	Uniq	
sub	String	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
family_name	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
given_name	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
email	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
phone_number	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
username	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
sid	String	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

It is possible to assign an LDAP directory attribute to a translator that converts the attribute from the format stored in LDAP to the required format in Blitz Identity Provider. For example, this can be useful if you need to process the `objectGUID` attribute from an Active Directory LDAP directory in Blitz Identity Provider so that the attribute is represented as a GUID string instead of a byte. This is [configured](#) (page 248) via a configuration file.

#### Computed attributes

To configure computed attributes in the Computed attributes block you should do the following actions:

- add a new attribute by clicking the `+Add attribute` link;
- specify the name of the computed attribute;
- specify the value type - data type format;
- specify the calculation rule of the attribute based on other attributes or assigning a constant value to it.

Example of rules:



- to create the `First name` and `last name` attribute from the stored attributes `family_name` and `given_name`, it is necessary to define the stored attributes `family_name` and `given_name`, and then set the computed attribute `full_name` with the computation rule - `${family_name} ${given_name}`.
- to create the attribute `e-mail domain` from the stored attribute `email` you need to define the stored attribute `email`, and then define the computed attribute `domain` and define its computation rule `${email###*@}`.

**Note:** You can find help on supported substitution string parameters [here](#)<sup>20</sup>.

#### Calculated attributes

If necessary, define the calculated attributes. To do it, specify their *name*, *value type* and also configure the *calculation rule* based on the stored attributes.

A computed attribute can be assigned a constant value.

##### Examples

Attribute name	Format	Expression	
			<a href="#">+ Add attribute</a>

#### Input value conversion rules

Input value conversion rules allow checking the correctness of the data input format and ensure that the data is saved in the correct format. Rules are specified using regular expressions. Each rule includes a regular expression that allows decomposition (splitting into parts) of the input value and a rule for saving the obtained parts (layout).

Example of solved tasks:

- to check if the `email` attribute contains the `@` character, you must specify a `^(.+).+(.+) $` decomposition expression and a  `${0-}` layout expression;
- to check the format of the `mobile` and save it in the format `+7(999)1234567`, you must specify the decomposition expression `^(\+?) ([78]?) ?\ (? ([0-9] {3}) \) ? ? ([0-9] {3}) [- ]? ([0-9] {2}) [- ]? ([0-9] {2}) $` and the composition expression `+7 ($ {3-}) $ {4-} $ {5-} $ {6-}`.

#### Rules for converting input values

These rules check the correctness of the data entry format and ensure that the data is saved in the correct format. Rules are defined using regular expressions.

##### Examples

Attribute name	Split rule	Conversion rule	
email	^(.+).+(.+) \$	\$(0)	<span style="color: red;">✘</span>
phone_number	^(+?) ([78]?) ?\ (? ([0-9] {3}) \) ? ? ([0-9] {3}) [- ]? ([0-9] {2}) [- ]? ([0-9] {2}) \$	+7 (\$ {3-}) \$ {4-} \$ {5-} \$ {6-}	<span style="color: red;">✘</span>

[+ Add rule](#)

<sup>20</sup> <http://tdp.org/LDP/abs/html/parameter-substitution.html>

## Output value conversion rules

These rules allow additional transformations to be performed on computed attributes. For example, only necessary groups can be extracted from an attribute with an array of user groups, or group values from the format `CN=name,DC=...` should be converted simply to CN names. Examples of settings for such conversion rules are shown in the figure below (the corresponding computed attributes must be *created* (page 43) beforehand).

### Rules for converting output values

These rules allow you to perform additional transformations with calculated attributes.

Attribute name	Split rule	Conversion rule	
			<a href="#">+ Add rule</a>

## Setting up attribute purpose

It is necessary to specify which attribute will be the identifier in the system. The identifier must be unique and not change over time.

**Note:** It is not recommended to change the base ID in the future, as all user settings are bound to it. Changing the base ID will lose two-factor authentication settings, registered security events, memorized user device lists, connections to external accounts, and user attributes stored in Blitz Identity Provider database.

You can also specify which attributes are used for special purposes:

1. An attribute used as a marker of account lockout. This attribute must have a value type of `Boolean`. Blitz Identity Provider supports locking out users stored in the LDAP directory. To use this feature, you must also *configure* (page 47) the appropriate attribute in the LDAP directory settings.
2. For example, the expression `${family_name} ${given_name} ${middle_name-}` allows an account (for example, under Users) to display the last name, first name, and patronymic name (if it's present).
3. Attributes used to store email addresses.
4. Attributes used to store mobile phone numbers.

Multiple attributes can be specified as e-mail and mobile phone attributes (e.g., for personal and work email address).

## Purpose of attributes

Specify which attribute is the identifier in the system. The identifier must be unique. It is not recommended to change the identifier.

You can also specify which attributes are used:

- to determine which accounts are locked. This attribute must be Boolean;
- as the email address;
- as the mobile phone number

Identifier	<input type="text" value="sub"/>
Lock marker	<input type="text"/>
Console username	<input type="text" value="{family_name-} {given_name-}"/>
Email	<input type="text" value="x@email"/>
Mobile phone	<input type="text" value="xphone_number"/>

Save

## Connecting attribute storages

### Types of storage

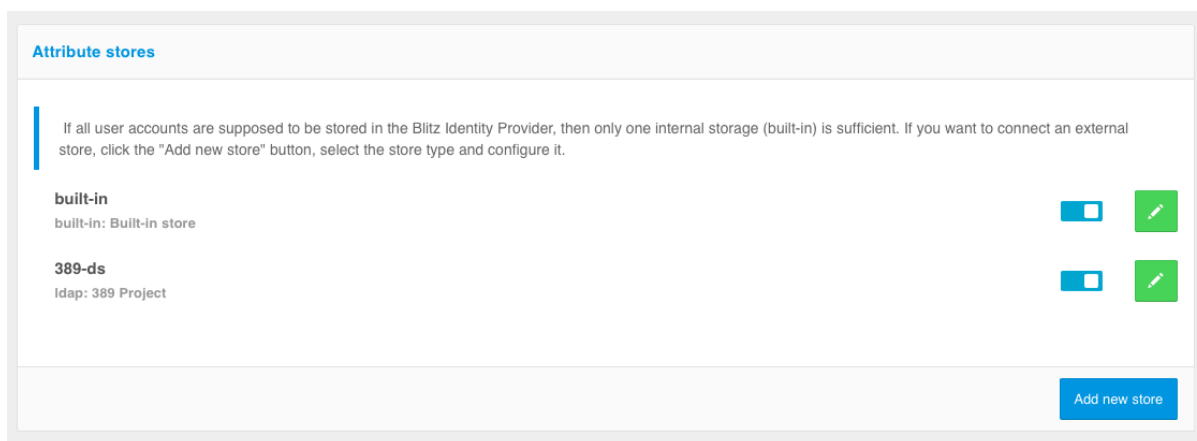
As user attribute storages Blitz Identity Provider allows you to use:

1. External (connected) storage. This can be:
  - LDAP repository - this can be any server that supports the LDAP (389 Directory Server, OpenLDAP, FreeIPA, etc.), as well as Microsoft Active Directory or Samba4;
  - other storage that requires special [REST services](#) (page 51) to be developed to connect to Blitz Identity Provider.
2. Internal Storage. All user attributes are stored in the Blitz Identity Provider database. If Couchbase Server is used as a DBMS, Blitz Identity Provider database can be used to store a small number of accounts. If PostgreSQL is used as a DBMS, any number of accounts can be stored.

Blitz Identity Provider requires configuring at least one storage and configuring [attributes](#) (page 42) to work correctly. By default, an internal configured storage and a number of attributes added.

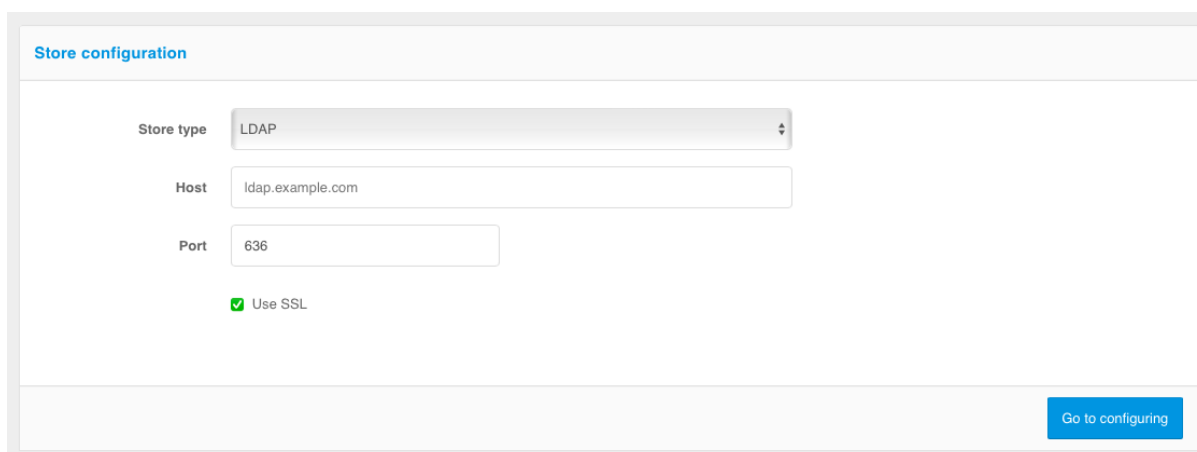
**Note:** Each user account is stored in one specific storage. Blitz Identity Provider allows you to configure and connect multiple storages, but it is recommended that you use one primary storage for operation. Use of a second storage should be decided based on the applicable data model. For example, the connected corporate Active Directory can store data of the organization's employees, and an additional LDAP storage can store data of specially registered "external" users (employees of partner organizations, freelancers, etc.).

To select and configure the storage to be used, you must first configure the attributes in the Data sources in Attribute storages section. The internal storage is configured by default. To add an external storage, click the Add new storage button, then specify the type of the external storage and configure the parameters of interaction with it. The created storages are disabled after creation - you should enable them using the toggle switch in the Attribute storages section.



If you do not use an internal storage, you can delete it. To do this, go to the properties of the corresponding external storage and click the Delete button.

Using multiple storages can solve the problem of users' logging records stored in different LDAP directories or in different branches of the same directory. For example, as a result of a merger between two companies, you can connect two directories to Blitz Identity Provider and provide user logins without configuring trust or building a meta-directory.



### Connecting storage via LDAP

If you are using an LDAP storage deployed in your organization as the source of user accounts, you must use the Data sources section of the admin console and perform the following steps to configure it:

- add a new storage, specify the following data:
  - type of storage to be added - select LDAP;
  - storage address;
  - port;
  - check the Use SSL box if a secure connection should be used.
- configure the LDAP-storage by configuring the following parameters:
  - storage description (optional);
  - whether the storage is to be used only for reading data or whether it is possible to write to it;
  - whether SSL connection should be used;

- whether DNS-calls balancing to the LDAP-storage is needed - to do this press the DNS-balances button and set the parameters Domain name, Port, Use SSL, Mode of operation, Cache storage time, ms;

**Note:** During DNS balancing, Blitz Identity Provider queries the DNS server for all connection addresses by the specified LDAP directory domain name. If more than one address is registered in DNS, then depending on the selected mode of operation Blitz Identity Provider establishes connection to the first available server (FAILOVER mode of operation), to a random server (RANDOM mode of operation) or to each server in turn (ROUND\_ROBIN mode of operation). The list of servers received from DNS is stored in Blitz Identity Provider cache for the time specified in the Cache storage time, ms setting.

- connection pool settings;
- specify the login and password of the user on behalf of whom the work with LDAP storage will be performed (this user must have the rights to read and write data), as well as the base DN - the partition of the directory with user accounts;

**Note:** It is acceptable to specify a user with read-only privileges if the storage is used only for reading.

- specify search settings - search depth and maximum number of returned accounts (this affects the number of users displayed in the Users section of the admin console).

### Connection properties

Identifier

Description

Read-only

---

### Connection settings

No balancing DNS-balancing

Host

Port

Use SSL

---

### Advanced connection properties

Connection timeout, ms <input type="text" value="3000"/>	Initial connections <input type="text" value="10"/>
Response timeout, ms <input type="text" value="3000"/>	Maximal connections <input type="text" value="10"/>

---

### Information to log on to server

The account should have the permissions to read the data from store

User(DN)

Password [Change value](#)

Base DN

---

### Search parameters

Search depth

The maximum number of records returned in search

You can further customize attribute matching rules and specify partitioning rules and attribute value conversion rules. This allows you to:

- give a system attribute a different name than its name in the LDAP directory. For example, if an attribute is specified as `sn` in the LDAP directory, but you want to use it as `family_name` in Blitz Identity Provider, select the attribute `family_name` and specify `n` as its name in LDAP. An example of this configuration is shown in the figure below;
- use special rules for writing attributes to a given LDAP directory. For example, if you want to store a cell phone in the format `+7 (999) 1234567` into an LDAP directory without brackets, then set a partitioning rule `^\+7\ ( ([0-9]{3}) \) ([0-9]{7}) $` and a conversion rule `+7${1-}${2-}` for the entry.
- use special rules for reading attributes from a given LDAP directory. For example, if an attribute with a cell phone number is specified in the `+79991234567` format in the LDAP directory, and Blitz Identity Provider uses the `+7 (999) 1234567` format, you can use the `^\+7 ([0-9]{3}) ([0-9]{7}) $` and the `+7 ($ {1-}) $ {2-}` conversion rule to read from the directory.

## Attribute matching rules

Configure the mapping rules if the attribute names or formats in the Blitz Identity Provider do not match the attributes definition in the LDAP directory. Both for reading and for writing you can specify split rules and rules for converting attribute values. This allows:

- to give the attribute in the system a different name that does not match its name in the LDAP directory. For example, if the attribute is specified in the LDAP directory as `sn` and Blitz Identity Provider needs to use it as `surname`, select the `surname` attribute and specify `sn` as its name in LDAP;
- use special rules for writing attributes to this LDAP directory. For example, if you want to save a mobile phone in the format `+7(999)1234567` to an LDAP directory without parentheses, then write the rule for splitting `^\+7\([0-9]{3}\)\($0-9]{7})$` and the `+7$(1-)}$(2-}` conversion rule.
- use special rules for reading attributes from this LDAP directory. For example, if the attribute with the mobile phone number is specified in the format `+79991234567` in the LDAP directory and the format of the Blitz Identity Provider is `+7(999)1234567`, then you can use the split rule `^\+7([0-9]{3})([0-9]{7})$` and the conversion rule `+7$(1-)}$(2-}`;

Attribute	Name in LDAP	Write		Read		
		Split rule	Conversion rule	Split rule	Conversion rule	
uid	uid					✖
surname	sn					✖
name	givenName					✖
mail	mail					✖
mobile	mobile					✖
username	login					✖

[+ Add attribute](#)

If an attribute *created earlier* (page 42) is not supposed to be stored in this storage, you can delete the attribute using the delete button. In this case, the value of the deleted attribute will be stored in Blitz Identity Provider database instead of in the external storage to be connected when creating/editing an account.

If you plan to use the account lockout feature, you must remove the attribute defined in the Data sources section as a lockout attribute from the attribute matching rules table.

If Blitz Identity Provider is used to register users, and the entry is written to this directory, you should specify the parameters for creating new users - the DN of the parent container within which the users will be created, and the system attributes related to the specifics of the storage\*\*\*.

**Note:** For example, `objectclass`, which defines the type of LDAP account to be created. For Microsoft Active Directory, `objectclass` must have the format `Array of string` and the value is `top, person`.

## Create new user parameters

To use the create users function you should enter the specific LDAP store parameters. When forming the values of the parameters you can use substitution strings of user attributes. List value can be specified using comma.

User DN

E.g. CN=\${mail},CN=users,DC=domain,DC=com

Name	Format	Value	
objectClass	Array of strings	top,blitz-schema-demo	✖

[+ Add attribute](#)

## Connecting to storage via REST

If an external database (not LDAP storage) is used as the source of user accounts, a connector must be developed to connect to it. The connector provides reading (or changing) the necessary data from the database and provides the data in the correct format as REST services to Blitz Identity Provider.

To configure interaction with REST services, perform the following steps:

- add a new storage, specifying the type of storage to be added - REST;
- specify storage description (optional);
- specify whether the storage is used only for reading data or whether it is possible to write to it;
- specify the maximum number of records returned by the search;
- specify the list of attributes available using REST services;
- specify the URLs of the following services:
  - service for user search;
  - service for receiving user data;
  - verification service for user login and password;
  - service for changing user password;
  - service for adding a new user;
  - service for changing user data;
  - service for deleting user.

The screenshot of the page with settings of connection to the storage using REST services is presented below.

**REST-service parameters**

Identifier:

Description:

Read-only:

The maximum number of records returned in search:

List of available attributes:

User attributes that can be used in REST-services

---

**REST-services URLs**

URL of service for user search:

HTTP request method: **GET**. Request parameter:  
 • **rq1** — request in **Resource Query Language (RQL)** format.  
 Response format: **200 OK**, user list in **JSON Array** format in **UTF-8** encoding.  
[Listing example](#)

URL of the user data retrieval service:

When specifying a URL you must use a substitution string for the user identifier - **\${id}**.  
 HTTP request method: **GET**.  
 Response format: **200 OK**, user data in **JSON** format in **UTF-8** encoding.  
 If the user is not found: **400 Bad Request**, error code **USER\_NOT\_FOUND** in **text/plain; charset=utf-8** format.  
[Listing example](#)

The following subsections describe the requirements for developing REST services that provide the necessary Blitz Identity Provider access to the accounts storage.



## Service for user search

The user search service must process requests using the GET method, where the search query is specified as the `rql` parameter. The query has the format [Resource Query Language \(RQL\)](#)<sup>21</sup> and must at least support the following operations:

- `limit` is the number of records to be return;
- `and` - simultaneous execution of search conditions;
- `or` - alternative execution of search conditions (for example, search by different attributes as a login);
- `in` - the occurrence of the attribute value in the list of values (for example, searching for linked accounts when logging in through an external identity provider);
- `eq` - equality condition check with the possibility to search by mask (for example, using a star (\*)).

For example, if only search as a login is configured in the Authentication section by `email` attribute, then the RQL-parameter sent during authentication will have the following form (where `test@mail.com` - data entered by the user as a login):

```
rql=and(eq(email,test@mail.com),limit(10))
```

If you are logging in using an external identity provider, and you want to find the accounts associated with the external account in the storage, the RQL parameter passed will be:

```
rql=and(in(sub,(7d5fd1d2-e171-4c85-8da6-00368863c396,2b78a2da-241c-4182-ba9b-  
↪d810cdb7aa70)),limit(10))
```

If `email` OR `sub` attribute search is configured as login, the passed RQL parameter will be of the form:

```
rql=and(or(eq(sub,test@mail.com),eq(email,test@mail.com)),limit(10))
```

The service should return the list of users and their data in JSON format in UTF-8 encoding. Attributes must be returned for each user:

- `id` - a user identifier in the connected database. This identifier implies that it will be unchangeable for this user;
- `attrs` - object with the list of returned user data. It is necessary to return those attributes, which are supposed to be used in the system and which are configured in the Data sources section.

Request example:

```
GET /users/search?rql=and(eq(sub,BIP*),limit(10)) HTTP/1.1  
Host: idstore.identityblitz.com  
Content-Type: application/json  
Cache-Control: no-cache
```

Response example:

```
[  
  {  
    "id": "ID123",  
    "attrs": {  
      "sub": "BIP123",  
      "given_name": "Ivan",  
      "family_name": "Ivanov",  
      "email": "ivanov@test.org",  
      "phone_number": "+79991234567"    }  
  }  
]
```

(continues on next page)

<sup>21</sup> <https://github.com/kriszyp/rql>

(continued from previous page)

```

    }
  },
  {
    "id": "ID456",
    "attrs": {
      "sub": "BIP456",
      "given_name": "Elena",
      "family_name": "Ivanova",
      "email": "ivanova@test.org",
      "phone_number": "+79997654321"
    }
  }
]

```

### Service for receiving user data

In some cases Blitz Identity Provider requests data of a particular user. The service for obtaining user data should process requests using the GET method, in which the `id` attribute - the internal user identifier in the connected database - is specified in the URL. When specifying the URL of this service in the admin console, you must use a substitution string for the user identifier - `${id}`, for example:

```
https://idstore.identityblitz.com/users/${id}
```

If the user is found, the service should respond 200 OK and return user data in JSON format in UTF-8 encoding. If the user is not found: 400 Bad Request, error code `USER_NOT_FOUND` in text/plain; charset=utf-8 format.

Request example:

```

GET /users/ID123 HTTP/1.1
Host: idstore.identityblitz.com
Content-Type: application/json
Cache-Control: no-cache

```

An example of a response if the user is found:

```

HTTP/1.1 200 OK
Date: Mon, 18 Jul 2016 12:28:59 GMT
Content-Type: application/json; charset=utf-8

{
  "id": "ID123",
  "attrs": {
    "sub": "BIP123",
    "given_name": "Ivan",
    "family_name": "Ivanov",
    "email": "ivanov@test.org",
    "phone_number": "+79991234567"
  }
}

```

Answer for the case when the user is not found:

```

HTTP/1.1 400 Bad Request
Date: Mon, 18 Jul 2016 12:28:59 GMT
Content-Type: text/plain; charset=utf-8

```

(continues on next page)

(continued from previous page)

```
USER_NOT_FOUND
```

### Verification service for user login and password

Verification service for user login and password must process POST requests, body should contain the following parameters (in the `application/x-www-form-urlencoded`):

- `id` - internal identifier of the user in the connected database;
- `password` - password.

If successful, the service should return a 200 OK response.

If the service cannot authenticate, it should return 400 Bad Request with one of the following errors:

- `INVALID_CREDENTIALS` - incorrect user login or password;
- `UNWILLING_TO_PERFORM` - user is locked;
- `INAPPROPRIATE_AUTHENTICATION` - user cannot be authenticated by password;
- `PASSWORD_EXPIRED` - user's password is expired.

Request example:

```
POST /users/bind HTTP/1.1
Host: idstore.identityblitz.com
Content-Type: application/x-www-form-urlencoded
Cache-Control: no-cache

id=ivanov&password=12345678
```

Response example (successful login and/or password):

```
HTTP/1.1 200 OK
Date: Mon, 18 Jul 2016 12:38:53 GMT
Content-Type: application/json; charset=utf-8
```

Response example (wrong login and/or password):

```
HTTP/1.1 400 Bad Request
Date: Mon, 18 Jul 2016 12:38:53 GMT
Content-Type: text/plain; charset=utf-8

INVALID_CREDENTIALS
```

### Service for changing user password

Service for changing user login and password must process POST requests, body should contain the following parameters (in the `application/x-www-form-urlencoded`):

- `id` - user identifier, obtained from the result of the user's password;
- `old_password` - old password;
- `new_password` - new password.

If successful, the service should return a 200 OK response.

In case of error, the service should return 400 Bad Request with one of the following errors:

- INVALID\_CREDENTIALS - user with the given ID and password is not found;
- UNWILLING\_TO\_PERFORM - user is locked;
- CONSTRAINT\_VIOLATION - new password does not correspond to the security policy.

The other errors returned should be similar to the procedure to verify the login and password.

Request example:

```
POST /users/changePassword HTTP/1.1
Host: idstore.identityblitz.com
Content-Type: application/x-www-form-urlencoded
Cache-Control: no-cache

id=ivanov&old_password=12345678&new_password=0987654321
```

Response example:

```
HTTP/1.1 400 Bad Request
Date: Mon, 18 Jul 2016 12:43:23 GMT
Content-Type: text/plain; charset=utf-8

CONSTRAINT_VIOLATION
```

### Service for adding a new user

Service for adding a new user must process PUT requests, body should contain the following parameters (in the application/json format):

- password - user's password (optional);
- attrs - user's attributes.

If successful, the service should return the user's data in JSON format in UTF-8 encoding.

If the password doesn't meet the security policy, the service should return 400 Bad Request with CONSTRAINT\_VIOLATION error.

If such a user already exists, the service should return 400 Bad Request with error USER\_ALREADY\_EXISTS and note that the user with this identifier already exists.

Request example:

```
PUT /users HTTP/1.1
Host: idstore.identityblitz.com
Content-Type: application/json
Cache-Control: no-cache

{
  "password": "*****",
  "attrs": {
    "sub": "ivanov@test.org",
    "email": "ivanov@test.org"
  }
}
```

Response example (user created):

```
HTTP/1.1 200 OK
Date: Mon, 18 Jul 2016 12:28:53 GMT
Content-Type: application/json; charset=utf-8
```

(continues on next page)

(continued from previous page)

```
{
  "id": "ID678",
  "attrs": {
    "sub": "ivanov@test.org",
    "email": "ivanov@test.org"
  }
}
```

Response example (account is already registered):

```
HTTP/1.1 400 Bad Request
Date: Mon, 18 Jul 2016 12:43:23 GMT
Content-Type: text/plain; charset=utf-8

USER_ALREADY_EXISTS:ivanov@test.org
```

### Service for changing user data

The service of changing user data should process requests using the POST method, the `id` attribute - the internal identifier of the user in the connected database - should be specified in the URL of the called service. When specifying the URL of this service in the admin console, you must use a substitution string for the user identifier - `${id}`, for example:

```
http://idstore.identityblitz.com/users/${id}
```

The body of the data change request contains the following parameters (in `application/json` format):

- `password` - the new value of the user's password (if the password is not sent, it must not change);
- `replaced` - new values of the user's attributes to be replaced or added;
- `deleted` - list of names of attributes to be deleted.

If successful, the service should return the user's data in JSON format in UTF-8 encoding.

If new password doesn't meet the security policy, the service should return 400 Bad Request with `CONSTRAINT_VIOLATION` error.

If such a user does not exist, the service should return 400 Bad Request with error `USER_NOT_FOUND`.

Request example:

```
POST /users/ID123 HTTP/1.1
Host: idstore.identityblitz.com
Content-Type: application/json
Cache-Control: no-cache

{
  "replaced": {
    "email": "ivanov@domain.org"
  },
  "deleted": ["family_name"],
  "password": "#####"
}
```

Response example:

```
HTTP/1.1 200 OK
Date: Mon, 18 Jul 2016 12:38:53 GMT
Content-Type: application/json; charset=utf-8

{
  "id": "ID123",
  "attrs": {
    "sub": "BIP123",
    "given_name": "Ivan",
    "email": "ivanov@domain.org"
  }
}
```

### Service for deleting user

The service of user account deletion should process requests using the `DELETE` method, the `id` attribute - the internal user identifier in the connected database - should be specified in the URL of the called service. When specifying the URL of this service it is necessary to use the substitution string for the user identifier - `${id}`, for example:

```
http://idstore.identityblitz.com/users/${id}
```

If successful, the service must return the response `200 OK`.

If a user does not exist, the service should return `400 Bad Request with error USER_NOT_FOUND`.

Request example:

```
DELETE /users/ID123 HTTP/1.1
Host: idstore.identityblitz.com
Content-Type: application/json
Cache-Control: no-cache
```

Response example:

```
HTTP/1.1 200 OK
Date: Mon, 18 Jul 2016 12:28:53 GMT
Content-Type: application/json; charset=utf-8
```

### Configuring internal storage

If you are using Blitz Identity Provider database as the source of your user accounts, you must take the following steps:

- add a new storage - specify the type of `BUILT-IN` storage to be added;
- specify the storage ID;
- provide a description of the storage;
- define whether the storage is read-only or not;
- specify the maximum number of accounts returned in search.

**Internal store properties**

Identifier

Description

Read-only

The maximum number of records returned in search

**Note:** If PostgreSQL is used as the DBMS, any number of accounts can be stored. If Couchbase Server is used as the DBMS, the internal storage can be used to store a small number of accounts.

## 2.2.2 Authentication

Authentication settings are defined in the section Authentication section of the Admin Console. The next sections contain information on how to work with those settings.

### How to work with authentication settings

Authentication settings are set in the section `:bdg-primary: Authentication` of the admin console. The settings are divided into tabs:

General settings

General settings that define user authentication

Password policies

Password policy settings

Security keys

Security key settings

First factor

Settings of authentication methods used for primary identification and authentication

Second factor

Settings of authentication methods used to confirm login

:Third factor

Optional tab, it is displayed only if it is configured to have an authentication method applied additionally after passing the checks of the first and second factors

Authentication methods are grouped by the first and second factor. To enable the authentication method, you must first configure it.

**Note:** The second factor is used to “strengthen” the first factor, e.g, the user in addition to the password is required to enter a special code, generated by mobile application

The set of methods may vary depending on the type of license used. To go to the method settings, click the button **Go to the method configuration** (when the method is initially configured) or link **Go to Settings** (to adjust the current preset settings).

**Authentication settings**

General properties Password policies Security keys **First factor** Second factor Impersonification

[Add an external authentication method](#)

<p><b>Username and password</b> <input checked="" type="checkbox"/></p> <p>To sign in user has to enter username and password</p> <p><a href="#">Go to properties</a></p>	<p><b>Domain authentication</b> <input type="checkbox"/></p> <p>The domain authentication is used to log in</p> <p><a href="#">Go to properties</a></p>
<p><b>Digital signature device</b> <input checked="" type="checkbox"/></p> <p>To sign in user has to use a smartcard or another digital signature device</p> <p><a href="#">Go to properties</a></p>	<p><b>Proxy-authentication</b> <input type="checkbox"/></p> <p>Authentication based on HTTP-headers set by the proxy-server. In particular, the header can include a certificate received upon the established SSL / TLS connection</p> <p style="text-align: center; background-color: #fff9c4; padding: 5px;"><b>Configure method</b></p>
<p><b>Login using external identity providers</b> <input checked="" type="checkbox"/></p> <p>The user will redirected to an external identity provider to sign in. The user is required to give consent to enable Blitz Identity Provider access to his data.</p> <p><a href="#">Go to properties</a></p>	<p><b>Login using temporary link</b> <input checked="" type="checkbox"/></p> <p>Login using a link. The link works for a limited period of time.</p> <p><a href="#">Go to properties</a></p>
<p><b>Authentication using a known device</b> <input checked="" type="checkbox"/></p> <p>The device is remembered after successful authentication. Users will log in automatically within a certain period</p> <p><a href="#">Go to properties</a></p>	<p><b>Authentication by code sent via SMS/push</b> <input checked="" type="checkbox"/></p> <p>Для первичного входа пользователю нужно ввести код из push-уведомления или SMS, отправленного на номер мобильного телефона</p> <p><a href="#">Go to properties</a></p>
<p><b>External method "test"</b> <input type="checkbox"/></p> <p>After a successful initial login, additional verification will be made using an external authentication service</p> <p><a href="#">Go to properties</a></p>	<p><b>Authentication by security key</b> <input checked="" type="checkbox"/></p> <p>Authentication is performed using WebAuthn or U2F security keys</p> <p><a href="#">Go to properties</a></p>
<p><b>QR code login</b> <input type="checkbox"/></p> <p>User must scan QR code using mobile app</p> <p style="text-align: center; background-color: #fff9c4; padding: 5px;"><b>Configure the method to use</b></p>	



**Authentication settings**

General properties Password policies Security keys First factor **Second factor** Impersonification

[Add an external authentication method](#)

**Hardware-generated one-time passwords (HOTP)**

After a successful initial login the user has to enter the code generated by a special device - the generator of one-time passwords

[Go to properties](#)

**Time-based one-time passwords(TOTP)**

After a successful initial login the user has to enter the code generated by a mobile application

[Go to properties](#)

**Duo push-authentication**

Confirm your login with the Duo Mobile mobile app - you must respond to a push notification

[Configure method](#)

**Log in from a known device**

It allows not to require the second factor authentication when logging in from a known device.

**Confirmation by code sent via SMS/push**

After a successful initial login the user has to enter the code sent to mobile phone

[Go to properties](#)

**Security key confirmation**

Authentication is performed using WebAuthn or U2F security keys

[Go to properties](#)

**Confirmation by the answer to the security question**

User should enter the answer to the security question

[Configure method](#)

Refer to the following sections for guidelines on how to configure each method. To enable or disable an authentication method, set the switch to the desired position.

## General settings

On the tab General settings of the section Authentication you can set:

- `Default authentication level`: specify First factor to require users to verify the first authentication factor only (except for users whose settings include the need to verify the second factor). Specify First and second factor to require users to verify the second authentication factor in addition to the first factor.
- `Session duration parameters`:
  - `Session inactivity timeout`: specify time in seconds within which a user session will remain active despite of the user inactivity (absence of transitions between different applications).
  - `Maximal session timeout`: specify maximum time in seconds within which a user session will remain active (regardless of whether there is any user action).

**Attention:** The duration of a user's SSO session can also be affected by the `blc` cookie validity period on the Blitz Identity Provider side. By default, the `blc` cookie validity period is 10800 seconds. If the maximum session duration exceeds this value, the user may be asked to log in again as soon as the cookie expires, even with an active SSO session. In this case, [make changes](#) (page 278) to the configuration file.

- `Logout screen display time (in seconds)`: time in seconds that indicates how long the logout screen will be shown to a user before they are automatically redirected to the application transition

page after the logout.

- Configure account memorization:
  - Account memorization is enabled by default. Disable it if necessary.
  - Account memorization: account memorization mode. Specify `Remember one account` to make each log-in by a new account in the browser overwrite the memorized log-in of the previous account or `Remember all accounts` so that each log-in by a new account adds another account to the list of memorized accounts in the browser.
  - Displayed username: specify how to form a username displayed on the login page as a regular expression, for example: `${family_name-} ${given_name-}`. This regular expression allows displaying the last name and first name of the user stored in the `family_name` and `given_name` attributes.
  - Displayed user ID: specify how to form an account ID displayed as the second line on the login page, as a regular expression, for example: `${email-$phone_number}`. This regular expression allows to display one of the contacts stored in the `email` or `phone_number` attributes (if both are present, `email` is displayed). You can use value masking when customizing. For example, the `${phone_number&maskInMiddle(3,3)}` rule will display the middle numbers of a phone number as `*`.
  - Show avatar: specify whether to display a user avatar on the login page.

**Authentication settings**

General properties Password policies Security keys First factor Second factor Impersonification

---

**Default authentication level** First factor

Specify the default authentication level for all users. If you choose the 'First and second factor' option, then all users have to pass two-factor authentication

**Session inactivity timeout** 600

Specify the session timeout (in seconds). Within this period the session will not expire in case of absence of user activity, i.e. no SSO events

**Maximal session timeout** 10800

Specify the session timeout (in seconds). After this period the session will expire even in case of user activity

**Logout screen timeout** 2

---

**Memorize Accounts**

Enabled

**Memorize Accounts** Remember all accounts

**User display name** \${family\_name} \${given\_name}

**User display ID** \${email-\$sub}

Show icon

Save

## Password policies

Password policies are configured on the Password policies tab of the Authentication section of the admin console.

**Authentication settings**

General properties **Password policies** Security keys First factor Second factor Impersonification

---

**Password complexity**

Minimum password length   
Enter the minimum number of characters in the password

Password dictionary    
Select a password dictionary file with each password on a new line. The file format must be txt.

Character groups   
Set the minimum number of character groups required in a password

Group name	Valid characters	Minimum characters
Numbers	<input type="text" value="[0-9]"/>	<input type="text" value="1"/>
Lowercase	<input type="text" value="[a-z]"/>	<input type="text" value="1"/>
Uppercase	<input type="text" value="[A-Z]"/>	<input type="text" value="1"/>
Special characters	<input type="text" value="[!@#\$%^&amp;*()+-?.,;:'\" { \}~&lt;='&gt;~\_]"/'/>	<input type="text" value="1"/>

---

**Reuse policy**

Disable old passwords, pcs.

Minimum password age in seconds.

Maximum password age in seconds.

Minimum number of characters that differ.

The following settings are available:

- The minimum password length is the number of characters in the password (at least 8 characters is recommended);
- Password dictionary - a text file containing a list of forbidden passwords is specified. Each password should be on a separate line. If large files are used, it is recommended to upload them directly to the server, and specify the path to the file in the `dicPath` setting in the `blitz.prod.local.idp.password-policy` settings block in the `blitz.conf` file.
- Character group - sets the minimum required number of character groups in the password. For each character group, you can set the settings in the character group table:
  - Valid characters - a regular expression is used to specify the set of characters of a group. For example, you can expand the allowed characters of numbers by changing the regular expression to the following

- [0-9·-9], you can expand the allowed character sets of letters - [a-zA-я] and [A-ZA-Я], add or remove the allowed special characters - [!@#\$\$%^&\* () +-?. , ; : ' ` " { } [ ] > < = ~ / \ \_].

- Minimum characters - how many minimum characters from the group must be used in the password that the group is considered to be involved in the password.
- Prohibit using old passwords - the setting specifies how many old passwords should be memorized to prevent entering a password from the history of used passwords when setting a new password.
- Minimum password lifetime - the minimum password lifetime, in seconds; until this time has elapsed, the user will not be allowed to set a new password. If this check should not be performed, the setting should be set to an empty value.
- Maximum password lifetime - the maximum lifetime of the password, in seconds; once this time expires, the user will be prompted to set a new password. If this check should not be performed, the setting should be set to an empty value.
- Minimum number of different characters - how many changed characters should be in the new password compared to the previous one (for cases when the user changes the current password to a new one). If this check should not be performed, the setting should be set to an empty value.

## Security key management

### Configuring security keys

Blitz Identity Provider allows you to use security keys (WebAuthn, Passkey, FIDO2, U2F) for identification and authentication. The [WebAuthn](#)<sup>22</sup> specification is used to interact with security keys.

The following key types are supported:

- **External keys** - are hardware devices in the form of USB keys or key fobs connected to PCs, tablets and phones via USB port, Bluetooth or NFC. The keys do not require drivers or plug-ins to be installed on the device - interaction with the keys is performed through the built-in capabilities of browsers.
- **Built-in keys** - Authentication mechanisms built into the device and operating system that support WebAuthn:
  - Windows Hello - you can sign in using Windows PIN, fingerprint verification or facial recognition;
  - Touch ID or password on your MacBook;
  - Touch ID or Face ID on an iOS cell phone or fingerprint verification or facial recognition in Android.

Security keys are configured on the tab Security Keys of the section Authentication of the admin console.

<sup>22</sup> <https://fidoalliance.org/fido2/>

**Authentication settings**

General properties Password policies **Security keys** First factor Second factor Impersonification

---

**Relying party entity name**   
Human-readable server name as WebAuthn Relying Party

**Relying party ID**   
This is ID as WebAuthn Relying Party. It must be origins effective domain.

**Signature algorithms**   
This member specifies the cryptographic signature algorithm with which the newly generated credential will be used

**Authenticator Attachment**   
If this member is present, eligible authenticators are filtered to only authenticators attached with the specified type.

**Credentials discovery mode**   
Whether the server needs to filter credentials based on the login entered by the user.

**Timeout, ms**   
Timeout value specifies a time, in milliseconds, that the server is willing to wait for the call to complete

**Displayed user name**   
Appears on the login page when user log in. Use substitution strings to form a name. E.g., "{family\_name-} {given\_name-}"

**Displayed user ID**   
Appears on the device when user log in. Use substitution strings to form an ID. E.g., "{email-}"

**Normal Counter Distance**   
During authentication, the server verifies that the passed signature counter matches the current counter on the server within an acceptable range. It is recommended to set the range to 1.

[Save](#)

The following settings are available:

- Authentication system name – it is necessary to set the name of the authentication system or application name suitable for displaying to users.
- Authentication system domain – must match the domain used by the authentication system or be a superior domain. Security keys will be issued to this domain.
- Signature algorithms – it is recommended to specify ES256 and RS256 algorithms as a minimum to work with Passkey, Windows Hello and most common hardware FIDO2 and U2F security keys.
- Limit Allowed Authentication Means – If “Not Selected” is selected, authentication means are not limited. If you select “Portable”, only hardware security keys (USB, Bluetooth, or NFC) will work. If you select “Built-in Platform”, only security keys built into devices (Windows Hello, Touch ID on MacBooks, Touch ID and Face ID on cell phones, and using your phone as a Bluetooth-enabled authentication tool) will work).
- Key Verification Mode – When “Browser detection” is selected, the user will be shown all security keys available on their device for the authentication system domain. When “Server Discovery” is selected, the user will be prompted for a login, and then only those keys that are available on the device and linked to the user’s account on the server will be shown.

- Wait Time – Specifies the time in milliseconds that the authentication system will wait for the browser to respond to a request to access the security key.
- Displayed user name – specifies the wildcard pattern according to which the name of the memorized user is displayed on the Security Key login page in the authentication system (relevant when using the “Server detection” mode).
- Displayed Account ID – Specifies a wildcard string pattern that displays the name of the security key to the user on the device.
- Normal Authentication Counter Shift – a setting that specifies that the authentication server will compare the authentication count on the device with the authentication count of the same key on the server and, if it differs by more than the number specified in the counter, will disallow the use of the security key (key cloning protection).

Blitz Identity Provider authentication server is configured as standard to trust all known root and intermediate certificates of the TPM modules, FIDO, as well as the current Apple and Google certificates required to verify the signature of FIDO2 and U2F attestation objects. If necessary, [скорректируйте](#) (page 265) allowed attestation certificates.

The use of security keys on the first and second factor is described in the following sections.

### Logging in via WebAuthn, Passkey, FIDO2

It is possible to use security keys (WebAuthn, Passkey, [FIDO2](#)<sup>23</sup>) to log in to Blitz Identity Provider.

To configure the login using security keys, you need to set the following settings on the tab First factor:

- Allowed attestation modes – using only `FULL` and `FULL_NO_ROOT` modes will increase security, but will not allow to use some keys for login, as well as Windows PIN code, because when registering such keys the attestation object comes without chipset or key manufacturer’s signature or using a self-signed key. The use of `SELF` mode allows an attacker to implement a man-in-the-middle attack to spoof the key at the time of registration, in case the user’s device is controlled by the attacker.
- Show method only to users who have bound a security key to the account – If Blitz Identity Provider has already identified the user, it already knows if security keys are configured for the user’s account. If security keys are not configured, you can configure that the user is not shown the login method using the security key.
- Equate the use of this method to the use of the first and second factor – if the option is enabled, logging in by security key will mean that the user has passed two-factor authentication.
- Правила соответствия – при входе по ключу безопасности пользователя просят ввести логин. Настройка правил соответствия позволяет указать правила поиска соответствия учетной записи введенному логину. Для найденной учетной записи будет запрошена проверка входа по ключу безопасности. Для создания правила используется строка подстановки: `${login}` – это строка, введенная пользователем в поле «логин». В результате, например, правило `email=${login}` означает, что строка, введенная пользователем, будет сравниваться с атрибутом `email` в хранилище данных.
- Attribute store selection rules – as in the case of sign in by login and password, by default the user search for authentication is performed in all active stores. In the Attribute store selection rules block you can [configure rules](#) (page 67), when executed, the user will be searched in a certain store.

<sup>23</sup> <https://fidoalliance.org/fido2/>

## Security key authentication

Allowed attestation modes

 FULL
  FULL\_NO\_ROOT
  SELF

Attestation mode is checked when the user registers the security key. FULL - checks for a root certificate in a trusted repository. FULL\_NO\_ROOT - a root certificate is not required. SELF - allows to accept a self-signed attestation statement. By default, only mode FULL is allowed.

- Show the method only to users who have linked a security key to their account. By default, the method is shown to all users.
- Equate the use of this method to the use of the first and second factor. If enabled, the sign in with WebAuthn or U2F security key is passed for a two-factor authentication

## Matching rules

For security key login to work correctly, specify how the login should be formed and which attribute in the data source it corresponds to. You can create a number of alternative rules to define the login.

To create a rule, use [substitution strings](#). For example, the rule `CN=${login}` means the string entered by the user will be compared to the `CN` attribute in the data store.

[View substitution strings](#)

<input type="text" value="sub"/>	=	<input type="text" value="\${login}"/>	<input type="button" value="✖"/>	
				<a href="#">+ add condition</a>
<b>OR</b>				
<input type="text" value="email"/>	=	<input type="text" value="\${login}"/>	<input type="button" value="✖"/>	
				<a href="#">+ add condition</a>
				<a href="#">+ add an alternative rule</a>



## Attribute store selection rules

By default, users are searched for authentication in all active attribute stores. In these block you can configure rules, in case of which the user will be searched only in specified attribute store.

Several alternate attribute store selection rules can be set. This will allow you to authenticate some users through one repository, and others through another one.

To create a rule, use [substitution strings](#).

[View substitution strings](#)
[Create rule](#)

## Login confirmation with WebAuthn, Passkey, FIDO2, U2F

It is possible to use security keys (WebAuthn, Passkey, FIDO2<sup>24</sup>, U2F) to log in to Blitz Identity Provider.

To configure login confirmation using security keys, you need to set the following settings on the tab Second factor:

- Allowed attestation modes – using only `FULL` and `FULL_NO_ROOT` modes will increase security, but will not allow to use some keys for login, as well as Windows PIN code, because when registering such keys the attestation object comes without chipset or key manufacturer’s signature or using a self-signed key. The use of `SELF` mode allows an attacker to implement a man-in-the-middle attack to spoof the key at the time of registration, in case the user’s device is controlled by the attacker.
- Show method only to users who have bound a security key to the account - If security keys are not configured, you can configure that the security key login confirmation method is not shown to the user.

**Security key authentication**

Allowed attestation modes: `x FULL` `x FULL_NO_ROOT` `x SELF`

Attestation mode is checked when the user registers the security key. `FULL` - checks for a root certificate in a trusted repository. `FULL_NO_ROOT` - a root certificate is not required. `SELF` - allows to accept a self-signed attestation statement. By default, only mode `FULL` is allowed.

Show the method only to users who have linked a security key to their account. By default, the method is shown to all users.

Cancel Save

## Logging in using login and password

To use the username and password login, the following matching rules must be specified - to determine how the given username relates to the users in the data store.

To create a rule, a wildcard string is used: `${login}` is the string entered by the user in the “login” field. As a result, for example, the rule `email=${login}` means that the string entered by the user will be compared to the `email` attribute in the data store;

<sup>24</sup> <https://fidoalliance.org/fido2/>



**Username and password**

To configure this method please specify the rules for matching the username with the attributes from store. You can specify several alternative rules to match the username. The username is not case-sensitive.

Use [substitution strings](#) to form the rules. E.g. the rule `CN=${login}` means that the username will be matched with the attribute `CN` from the data store.

[View substitution strings](#)

sub = \$(login)   [+ add condition](#)

OR

email = \$(login)   [+ add condition](#)

OR

phone\_number = \$(login)   [+ add condition](#)

OR

username = \$(login)   [+ add condition](#)

[+ add an alternative rule](#)

In the log-in settings, it is possible to enable the [password policy check](#) (page 245). The password entered by a user will be checked against the password policy at log-in time. If the password does not meet the policy requirements, the user can set a new password or skip this step.

To configure password validation against the password policy at login, you must:

- select the `Always check the user's current password against the password policy` option or enter the name of some header in the `Check if HTTP header is present` field (in this case, if the HTTP request contains the specified header with the `true` value, the current user password will be checked against the password policy);
- the option `Allow the user to skip changing a password that does not comply with password policies` allows the user to refuse to change the password when logging in;
- specify the number of failed attempts for temporary blocking. After the specified number of failed attempts, the user will be temporarily blocked from using this authentication method;
- duration of the temporary blocking (in minutes).

#### Password compliance with password policy

Allow a user to skip change password not matching the password policy

Always check the current password against the password policy

**Check if HTTP header is present**

If the HTTP request contains the specified header with value true, then the current password will be checked against the password policy

**Number of unsuccessful attempts for temporary blocking**

After the specified number of unsuccessful attempts the account will be temporarily blocked for using this authentication method

**Duration of temporary blocking**

Defines the duration of the temporary blocking in minutes. After this period the user will be able to use this authentication method

You can control the password protection in the login settings. When the protection is enabled, the password verification is slowed down. After entering the password, the user will wait for the verification for the specified period `Delay time` (in seconds).

Administrator can select the following protection modes in the `Protection` setting:

- Automatic mode at system and user level - protection will be enabled for all users if the percentage of unsuccessful authentications exceeds the `Enable system protection at threshold`, and disabled if the percentage of unsuccessful authentications falls below the `Disable system protection at threshold`;
- Automatic mode at user level - the protection will be triggered for users for whom the number of unsuccessful password checks exceeds the number specified by the `Enable user protection at threshold` setting;
- Authentication delay for all users - protection will be enabled for all users;
- Disabled - the protection will be disabled.

System protection activation threshold and System Protection Disable Threshold parameters are set in percentages corresponding to the percentage of unsuccessful authentications in the total number of authentication attempts.

An example of how to configure password protection is shown below.

#### Brute force attack protection

When the protection is enabled, the authentication process slows down. In this case, after entering the password, the user will wait for the result for the period defined by the "Delay time" setting. There are two automatic protection modes:

- at the system level. It is turned on if the percent of unsuccessful authentications reaches a certain threshold (the setting "Enable system protection at threshold");
- at the user level. It is turned on if the user enters a certain number of invalid passwords in succession (setting "Enable user protection at threshold").

Protection	Disabled
Delay time in seconds	10
Enable user protection at threshold	5
Enable system protection at threshold	40
Disable system protection at threshold	30

To complicate automatic password mining, you can enable the `Proof of work performance` setting in Blitz Identity Provider. Then each time a user logs in with a username and password, the user's browser will have to perform a computationally complex task. If you fail to provide a solution, provide an incorrect solution, or provide a solution at the wrong time, Blitz Identity Provider will return an error. As a result, it will be impossible to know if the username and password are correct.

## Proof of work

There will be quite a time-consuming work on the browser side for each password verification. The result of this work will be checked by the server at the same time as the password is checked.

Check proof of work

Prompt only when HTTP header is present   
Proof of work will be requested if the HTTP request has a header value of 1

Work difficulty index   
The factor from 1 to 160 Bit. Each Bit increases the complexity by 2 times.

Maximum solution time   
Maximum time in seconds it takes for the browser to send a solution. If the value is not specified, the problem is expected to be solved in 1800 seconds.

The following can be configured in the `Proof of work performance` settings block:

- enable the `Request proof of completion of work` setting.
- if necessary, set the `Request only if HTTP header is present` setting - this is useful if you want to allow autotests to log in by password without having to pass the check. In this case, on the web server it is necessary to set the header from this setting for user requests, and not to set the header for requests coming from autotests.
- set `Work complexity index` - the coefficient value from 1 to 160 bits is set. Each bit increases the complexity by 2 times. The recommended value is 15 bits.
- `Maximum decision time` - time in seconds, in which the browser should send the result of the work. If the value is not specified, the task is expected to be solved in 1800 seconds. The time is counted from the moment the server generates the task at the moment of displaying the login page.

After setting the setting, it is recommended to press the `Test calculation` button before saving to get a rough idea of the run time on the current unit.

In the Rules for selecting an attribute repository block you can configure the rules, when executing which the user will be searched only in the specified store. By default, users for authentication are searched in all active attribute storages. You can specify several alternative storage selection rules. This will allow authenticating some users through one repository and others through another. Substitution strings are used to create a rule.

For example, in the screenshot below, it is configured that when a login is requested by an application with the `test_app` identifier, the user's login and password will be checked against the `test_db` repository. Login to all other applications will be performed through the `main` repository.

**Attribute store selection rules**

By default, users are searched for authentication in all active attribute stores. In these block you can configure rules, in case of which the user will be searched only in specified attribute store.

Several alternate attribute store selection rules can be set. This will allow you to authenticate some users through one repository, and others through another one.

To create a rule, use [substitution strings](#).

[View substitution strings](#)

**`\${login}`** data entered by user in the "login" field

**`\${\_rpId}`** application identifier (client\_id) in which the user is logging in

Attribute store	Matching rule		
main	<input type="checkbox"/> not	`\${_rpId}` ^(.*)\$	<input type="button" value="x"/>
			<a href="#">+ Add alternative condition</a>
test_db	<input type="checkbox"/> not	`\${_rpId}` test_app	<input type="button" value="x"/>
			<a href="#">+ Add alternative condition</a>
			<a href="#">+ Add rule</a>

## Logging in with electronic signature tool

### Configuring the authentication method in the Admin console

When using an electronic signature tool for authentication, you must:

- in the Certificates setting block load the certificates of the certification authorities, confirming the authenticity of electronic signature key certificates and configure interaction with the external electronic signature verification service.
- configure in the Compliance rules block the parameters of matching a user account in the storage by its attributes from an electronic signature certificate. Matching rules use substitution strings. For example, the `cn=${SUBJECT.CN}` rule means that the `SUBJECT.CN` attribute of the certificate will be compared to the `cn` attribute in the data store. It is possible to specify multiple conditions at the same time, as well as to specify alternative rules.

When configuring electronic signature login, you can also specify:

- whether this method should be used as the first and second factor. If yes, a user authenticated by electronic signature will be considered to have passed two-factor authentication (see the figure below for an example setting);
- whether to check the validity of the certificate. In this case, Blitz Identity Provider will use the revocation list distribution point (CRL) specified in the certificate to check if the certificate has been revoked. To activate this feature, check the checkbox `Verify that the user's certificate has not been revoked`;
- whether to create (register) an account at the first login by e-signature. In this case, if the user is not found by certain matching rules, the user will be prompted to register an account. To enable this feature, you should check the checkbox `Create an account if the user is not found by the electronic signature certificate` and configure the user registration rules - how to fill in the attributes in the repository from the certificate attributes. You should use substitution strings to set the rules. For example, the `email=${SUBJECT.E}` rule means that the `email` attribute will store the e-mail from the user's electronic signature certificate.

### General properties

Equate the use of this method to the use of the first and second factor. If enabled, the sign in with a digital signature device is passed for a two-factor authentication

Verify that the user certificate is not revoked

Cancel Save

### Matching rules

To configure this method please specify the rules for matching the field from the certificate with the attributes from store. You can specify several alternative rules. Use [substitution strings](#) to indicate the certificate fields. E.g. the rule `CN=${SUBJECT.CN}` means that field `SUBJECT.CN` of the certificate will be matched with the attribute `CN` from the data store.

[View substitution strings](#)

email

=

\${SUBJECT.E}

×

+ add condition  
+ add an alternative rule

Cancel Save

### Creating an account

If the user is not found while signing on using a digital signature, an account for this user can be created. Enable this function and specify how the Blitz Identity Provider attributes should be taken from the certificate attributes. Use the [substitution strings](#). For example, the rule `mail=${SUBJECT.E}` means that the mail attribute should be read from the email field of the certificate.

[View substitution strings](#)

Create an account if the user is not found using the digital signature certificate

Attribute	Rule	Master	
(not configured)		<input type="checkbox"/>	<div style="background-color: #f00; color: white; padding: 2px 5px; border-radius: 50%; display: flex; align-items: center; justify-content: center;">×</div>

+ Add attribute

Cancel Save

### Certificates

Load the Certification authority (CA) certificate that verifies the validity of user certificates.

Enter the path to the certificate for upload

Browse...
Load

Serial number	Issued	Issued by	Valid
850150393492941037372081	CN="ООО \\"КОМПАНИЯ \\"ТЕНЗОР\\"", O="ООО \\"КОМПАНИЯ \\"ТЕНЗОР\\"", OU="Удостоверяющий центр, STREET="Московский проспект, д. 12", L=г. Ярославль, ST=76 Ярославская область, C=RU, OID.1.2.643.3.131.1.1=#120C303037363035303136303330, OID.1.2.643.100.1=#120D31303237363030373837393934, EMAILADDRESS=ca_tensor@tensor.ru	CN=Минкомсвязь России, OID.1.2.643.3.131.1.1=#120C303037373130343734333735, OID.1.2.643.100.1=#120D31303437373032303236373031, O=Минкомсвязь России, STREET="улица Тверская, дом 7", L=г. Москва, ST=77 Москва, C=RU, EMAILADDRESS=dit@minsvyaz.ru	from 9/4/19 to 9/4/34

## Using and updating the plug-in

A special plugin - Blitz Smart Card Plugin - is used on users' computers for correct operation of the e-signature login. When logging in by e-signature for the first time, the user will be prompted to install the plugin. After downloading the file and running it, the user should go through all the steps of the plugin installation. When logging in again from this device, the plugin will not need to be installed again.

Blitz Identity Provider comes with a version of the plugin that allows you to work with electronic signatures as an authentication method.

If you need to update the Blitz Smart Card Plugin version, you should replace the plugin distributions - they are located in the `assets` directory with the Blitz Identity Provider installation, in the `assets.zip` archive. The structure of the archive is as follows:

```
plugins/sc/deb/BlitzScPlugin.deb
plugins/sc/rpm/BlitzScPlugin.rpm
plugins/sc/win/BlitzScPlugin.msi
plugins/sc/mac/BlitzScPlugin.pkg
plugins/sc/mac/BlitzScPlugin-10.14.pkg
...
```

You need to unzip the `assets.zip` archive, replace the files with the plugin distribution and zip the files back to `assets.zip`.

## Logging in via external identification services

The list of available external identity services depends on the edition of Blitz Identity Provider and the options purchased.

Logging in using the following external identity providers is possible:

- Apple ID;
- Facebook<sup>2</sup>;
- Google;
- identity providers running OpenID Connect;
- identity providers running SAML.

Connections to external identity services must be *preconfigured* (page 109) in the Admin Console on the tab Social login providers.

In the settings section Logging in via external identification services you must select which of the configured identity providers should be used for logging in.

Login using external identity providers

To add and configure the identity providers use the appropriate console section «[identity providers](#)».

Provider name	Provider ID	Provider type	
Google	google_1	google	<input checked="" type="checkbox"/>
Apple	apple_1	apple	<input checked="" type="checkbox"/>
Yandex	yandex_1	yandex	<input checked="" type="checkbox"/>
Mail ID	mail_1	mail	<input checked="" type="checkbox"/>
Facebook	facebook_1	facebook	<input checked="" type="checkbox"/>
VK	vk_1	vk	<input checked="" type="checkbox"/>
Odnoklassniki	ok_1	ok	<input checked="" type="checkbox"/>
ESIA	esia_1	esia	<input checked="" type="checkbox"/>
Digital Profile ESIA	esiadp_1	esiadp	<input type="checkbox"/>

### Logging in with proxy authentication

Proxy authentication (authentication by proxy server) is performed with the data sent in HTTP headers.

---

**Important:** When proxy authentication is enabled, Blitz Identity Provider only identifies the user, while authentication (as a result of certificate verification) is performed by the proxy server. Enabling this authentication method is acceptable when all users access Blitz Identity Provider through the proxy server.

---

For this method to work correctly you need to specify:

- required HTTP headers - list of HTTP headers that must be present to pass user proxy authentication,
- HTTP header with user certificate (optional parameter) - header containing x.509 user certificate,
- matching of HTTP header values and user identity data in the attribute store.

It is possible to configure mapping of attributes of the certificate passed in the HTTP header and user data to the storage.

An example of proxy authentication login settings is shown below:

**Proxy-authentication**

To use this authentication method, make sure the proxy server must be configured to transmit user credentials in the HTTP headers. This method is applied automatically if HTTP-headers include necessary identity data. If headers are not found other authentication methods are used

---

**HTTP-headers**

Required HTTP-headers    
 To add a HTTP-header enter it and press Enter

Specify the HTTP-headers names that must be present for user authentication. If the headers are not specified, the authentication will be possible for any set of headers

HTTP-header with user certificate    
 The HTTP-header that contains user certificate. If specified, the identification based on certificate fields is possible

---

**Matching rules**

To configure this method please specify the rules for matching the HTTP-headers with the attributes from store. You can specify several alternative rules. Use substitution strings to indicate the HTTP-headers. E.g. the rule `CN=${HTTP_X_SSL_CLIENT_CN}` means that the header `HTTP_X_SSL_CLIENT_CN` will be matched with the attribute `CN` from the data store. If a certificate is transmitted within a specific header, you can configure rules to match the certificate fields with attributes in the data store using [substitution strings](#).   
[View substitution strings for X509 certificates.](#)

=

[+ add condition](#)  
[+ add an alternative rule](#)

### Logging in using operating system session

The operating system session logon method allows users to avoid additional identification and authentication in Blitz Identity Provider if they have previously logged on to the organization's network from their PC and have been identified and authenticated in the operating system (logged on to the network domain). Such users will have end-to-end identity access to all applications connected to Blitz Identity Provider.

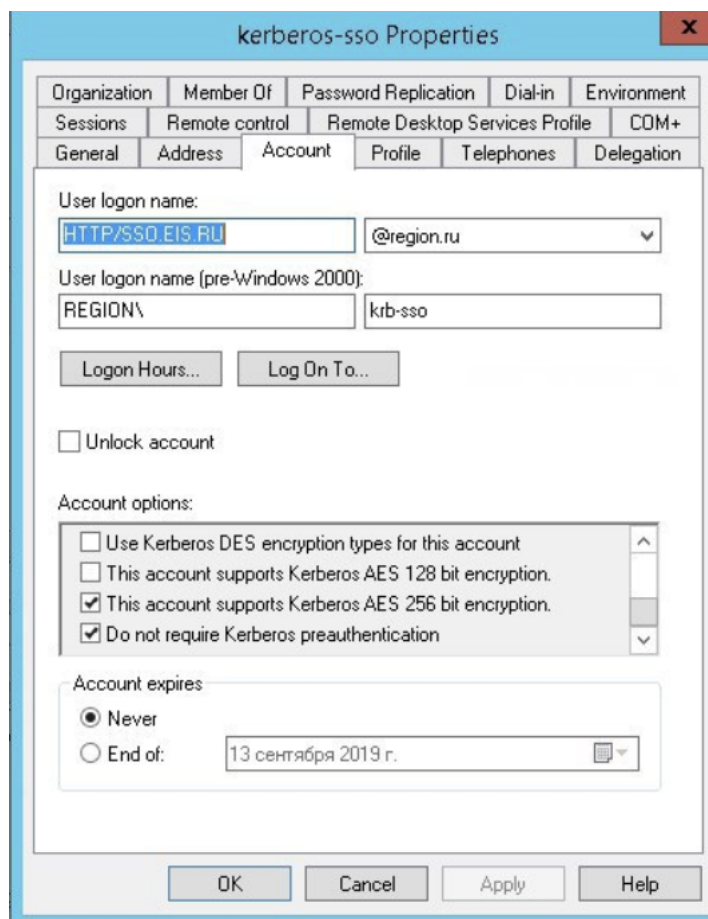
To log in using an operating system session, an organization must have a Kerberos server deployed (alone or as part of the organization's domain controller) and configured as described below.



## Domain controller (Kerberos server) configuration

In the domain controller you need to register an account for Blitz Identity Provider server. For the created account, on the Account page in the Account options block of the domain controller snap-in, enable the settings `User cannot change password` and `Password never expires`.

Also note the options `This account supports Kerberos AES 256 bit encryption` and `disable pre-authentication` `Do not require Kerberos pre-authentication`.



In the Group Policy Management snap-in, configure the `Configure encryption types allowed for Kerberos` policy by specifying the following possible values: `RC4_HMAC_MD5`, `AES128_HMAC_SHA1`, and `AES256_HMAC_SHA1`.

Example of configuration:



## Settings in Blitz Identity Provider admin console

It is necessary to go to the Authentication section in the management console to the settings of the login method Logging in by operating system session. In the opened window, load the previously generated keytab file. The SPN name will be set automatically in accordance with the uploaded file.

Based on the results of the keytab file download, information about the corresponding Kerberos service will be displayed.

If necessary, you can:

- delete the loaded keytab file;
- load more keytab files if you connected Blitz Identity Provider to more than one domain controllers.

Настройки субъекта службы Kerberos (SPN)

Загрузите файл таблицы ключей (keytab), сгенерированный для субъекта службы Kerberos `HTTP/bip-dev1.reaxoft.loc`

Укажите путь к файлу ключей для загрузки

Обзор Загрузить

Файл ключей SPN	Субъект службы Kerberos
bip-dev.keytab	HTTP/bip-dev1.reaxoft.loc@LAB.REAXOFT.LOC

Next, you need to define the matching parameters for the Kerberos token (TGS) and the account in Blitz Identity Provider.

Binding rules

To configure domain authentication (login using OS session) select which attributes from user store match the user name and domain name from the current OS session. You can create several alternative rules.

To create rules use [substitution strings](#). E.g. rule `userPrincipalName=${username}@${domain}` means that username with domain name will be matched with attribute `userPrincipalName` in user store.

[See substitution strings](#)

sAMAccountName = \${username}

+ add condition

+ add an alternative rule

Cancel Save

For example, you can specify that the user ID (username) received from the Kerberos token must match the `sAMAccountName` attribute received from the LDAP directory (Microsoft Active Directory).

The next step is to set the delay parameters for the login method using an operating system session.

Additional properties

Delay before using method   
The number of seconds during which the user can switch to a different authentication method

The waiting time of token receipt   
Number of seconds to wait for token receipt. At the end of the period an error message is returned

Cancel Save

Blitz Identity Provider provides two possible scenarios for using the operating system session:

**Basic Scenario.** Users log in to the operating system, and thereafter must end-to-end log in to all applications connected to Blitz Identity Provider. Providing users with the ability to log into applications under a different account is not required. In this case, you should set the `Delay time before method start` to 0 seconds. When the application is accessed, an end-to-end login will be attempted immediately through the operating system session.

**Additional scenario.** Users are not always able to log on to the operating system domain, or users in some cases need to be able to log on to applications under a different account than the one they used to log on to the domain. In this case, the `Delay time before method start` should be set so that the user has enough time to be able to cancel automatic login using an operating system session.

Waiting of token receipt should be set sufficient to allow the Kerberos server to respond to Blitz Identity Provider. Usually 5 seconds is sufficient.

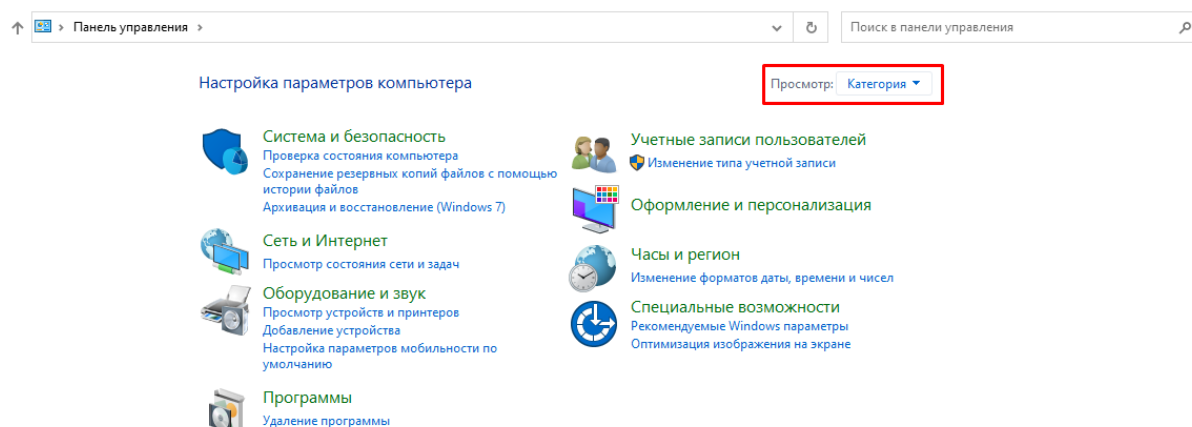
As in the case of login by login and password, by default the user search for authentication is performed in all active storages. In the Rules for selecting an attribute repository block you can configure the rules, when executed, the user search will be performed in a *certain storage* (page 67).

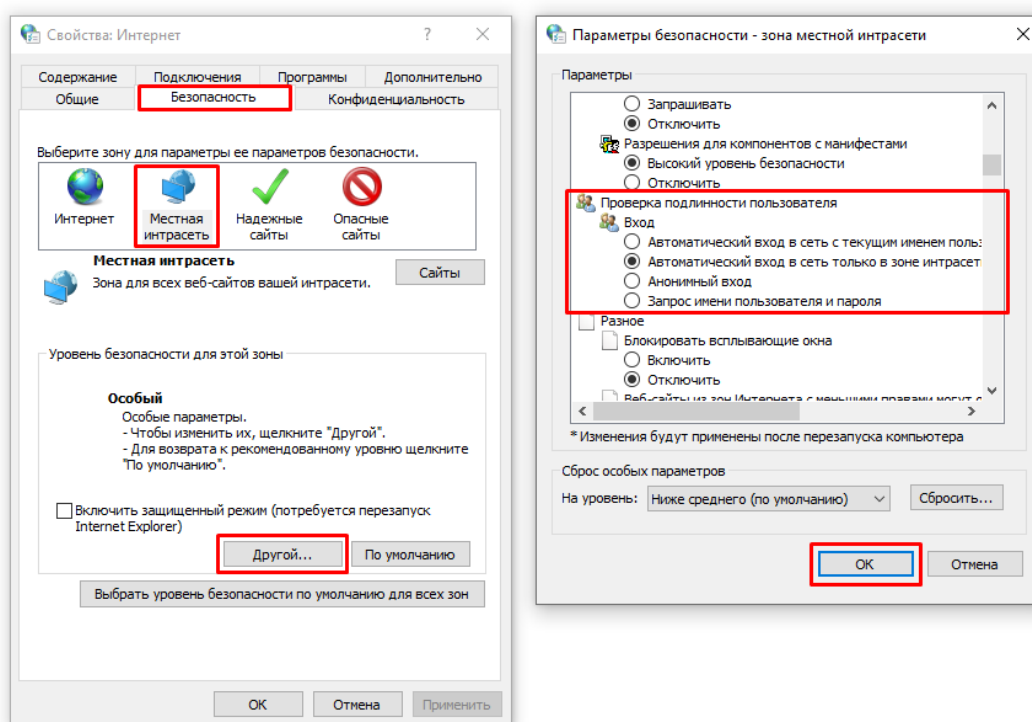
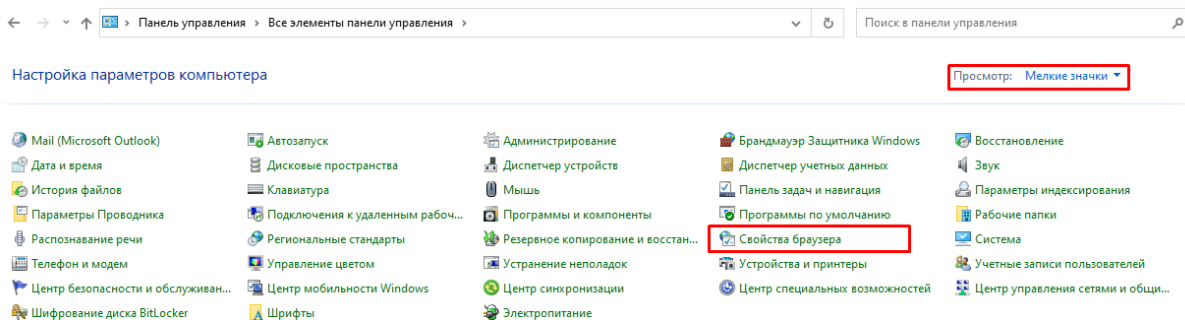
## Users' browsers configuration

Depending on the browser used by the user, it may be required to additionally configure it to support Kerberos authentication.

For Windows browsers, you need to set the following settings:

- open Start → Control panel, change the viewing option from `Category` to `Small icons`, select Browser properties in the opened settings;
- in the new window, select Security → Local intranet and click Websites. In the window that opens, click Additional and add Blitz Identity Provider site to the list of Local intranet sites by clicking Add;
- in the Properties: Internet → Security → Local intranet window, click the Other... button. In the window that opens, find the User authentication → Login setting. Set it to Automatic network login only in the intranet area.





You can choose not to make the above settings for the Windows operating system and, as an alternative, to be able to log in by operating system session in the Google Chrome browser, then you can start the browser with the following startup settings:

```
Chrome.exe -auth-server-whitelist="idp.domain.ru" -auth-negotiate-
->delegatewhitelist="idp.domain.ru" -auth-schemes="digest,ntlm,negotiate"
```

Where as `idp.domain.ru` you need to specify the URL of Blitz Identity Provider site.

You can also set the following settings in the Windows registry to run the Google Chrome browser without startup options.

```
Windows Registry Editor Version 5.00
[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Google]
```

(continues on next page)

(continued from previous page)

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Google\Chrome]
"AuthNegotiateDelegateWhitelist"="idp.domain.ru"
"AuthSchemes"="basic,digest,ntlm,negotiate"
"AuthServerWhitelist"="idp.domain.ru"
```

For Mozilla Firefox, you need to set the following settings (for any operating systems):

- enter `about:config` in the browser address bar and press `Enter`. In the next window, enter `network.nego` in the `Filters` field. Double-click on the `network.negotiate-auth.trusted-uris` entry found and set it to the URL of the site with Blitz Identity Provider, for example, `idp.domain.ru`. When specifying addresses, you can use an asterisk (\*) and specify multiple URLs separated by commas, for example: `https://*.idp.domain.ru,http://*.idp.domain.ru`. Close the pop-up window with the `OK` button.
- double-click on the `network.negotiate-auth.delegation-uris` entry you found and set it to the URL of the site with Blitz Identity Provider, for example, `idp.domain.ru`. When specifying addresses, you can use an asterisk (\*) and specify multiple URLs separated by commas, for example: `https://*.idp.domain.ru,http://*.idp.domain.ru`. Close the pop-up window with the `OK` button.
- open the `network.auth-sspi` parameter, set its value to `true`;
- restart the browser.

For Google Chrome in macOS and Linux, you need to run Google Chrome in a special way:

```
"/Applications/Google Chrome.app/Contents/MacOS/Google Chrome" --args --auth-
↪server-whitelist="idp.domain.ru" --auth-negotiate-delegate-whitelist="idp.domain.
↪ru"
```

Where as `idp.domain.ru` you need to specify the URL of Blitz Identity Provider site.

No separate configuration is required for Apple Safari in macOS.

### Blitz Identity Provider application launch settings

Users may have problems logging in by operating system session if they use Internet Explorer browser and if their account is included in many security groups in the domain, or if the DN of the account is long enough. To avoid this situation, it is necessary to set a special JAVA parameter when launching the `blitz-idp` authentication service application that defines a large allowable HTTP header size. To do this, edit the `/etc/default/blitz-idp` file. Add a key to the `JAVA_OPTS` parameter:

```
-Dakka.http.parsing.max-header-value-length=16K
```

### Web Server configurations

Users may have problems logging in by operating system session if they are using the Internet Explorer browser, and if their account is included in many security groups in the domain, or if the DN of the account is long enough. To avoid this situation, you must adjust the web server settings that determine the allowable size of header buffers.

Recommended buffer values for nginx are given below:

```
proxy_buffer_size 16k;
proxy_buffers 4 16k;
proxy_busy_buffers_size 16k;
client_body_buffer_size 16K;
```

(continues on next page)

(continued from previous page)

```
client_header_buffer_size 16k;
client_max_body_size 8m;
large_client_header_buffers 2 16k;
```

### Debugging operating system session login problems

If the operating system session login still does not work for users with the settings made, it is recommended to run the following command on the user's computer at the command line:

```
klist
```

If the command successfully returns TGS credentials for the SPN configured for Blitz Identity Provider, then you should check the correctness of the settings on the user's browser side and in Blitz Identity Provider. If TGS credentials for Blitz Identity Provider are missing, you can request them using the following command (you must specify the correct SPN and company domain name):

```
klist get HTTP/idp.company.ru@DOMAIN.LOC
```

If the command does not return the received TGS credentials, then we need to check if the settings on the Kerberos server are correct.

### Logging in with email

Blitz Identity Provider allows logging in using email as the first authentication factor. In this case, for logging in a user is required to enter the code sent to their email address. To configure the method, follow the steps below.

#### Step 1. Add the method to blitz.conf

To make the Authentication by email method appear on the First factor tab, do the following:

1. Open the `/usr/share/identityblitz/blitz-config/blitz.conf` file.

```
sudo vim /usr/share/identityblitz/blitz-config/blitz.conf
```

2. In the first list of the `blitz.prod.local.idp.login.factors` settings block, add a new block with the email method:

```
"login" : {
  "factors" : [
    [
      ...
    ],
    [
      {
        "enabled" : false,
        "method" : "email"
      },
      ...
    ]
  ],
  ...
}
```

3. Restart the services.

```
sudo systemctl restart blitz-idp blitz-console blitz-recovery
```

## Step 2. Configure the method in the console

In the admin console, do the following:

### 1. On the Authentication by email tab, configure the following settings:

- Method of account identification – specify a regular expression. For example, the `email=${login}` rule means that the value entered by a user in the login form will be matched with the `email` attribute.
- Length of the confirmation code.
- Code validity period.
- Number of attempts per log-in to enter the confirmation code.
- Total number of attempts (number of code sends and code entry attempts, after which this authentication method will be temporarily blocked for the user).
- Blocking time when attempts are exceeded (in minutes).
- Sending method: specify the attribute as an expression that indicates where a user's email address is stored, for example, `${email}`.

For the initial login, the user needs to enter the code from the message sent to the email address

[View substitution strings](#)

email = \${login} + add condition

OR

sub = \${login} + add condition

[+ add an alternative rule](#)

---

Confirmation code parameters

Length   
Confirmation code character limit

Expiration time   
The number of seconds before the confirmation code will be invalid. Sending a new code is required

Number of attempts for 1 login   
The number of failed attempts to enter the verification code in a single login attempt. If the number of attempts is exceeded, a new code must be sent.

Total attempts   
The total number of verification code submissions and verification code attempts that will cause the authentication method to be temporarily blocked

Blocking time when attempts are exceeded, in min.   
During the specified time, the authentication method will be unavailable to the user

---

Sending options

Attribute with contact   
Expression that will generate the email address to send the confirmation code

- Set the attribute store selection rule to search for a user-entered email address.



## Attribute store selection rules

By default, users are searched for authentication in all active attribute stores. In these block you can configure rules, in case of which the user will be searched only in specified attribute store.

Several alternate attribute store selection rules can be set. This will allow you to authenticate some users through one repository, and others through another one.

To create a rule, use [substitution strings](#).

[View substitution strings](#)

[Create rule](#)

2. Enable the Authentication by email method on the Authentication -> First factor tab.
3. Configure the Blitz Identity Provider connection to the [SMTP service](#) (page 152).

## Logging in with confirmation codes

You can use push notifications sent to the mobile app, or SMS as the first factor of authentication.

**Attention:** If a user does not have a mobile phone number, they will not be able to use the login verification via SMS.

To use the confirmation codes, you must:

- configure and enable the authentication method Authentication by code sent via SMS/push. You need to configure:
  - way to identify an account - specify a regular expression. For example, the `phone_number=${login}` rule means that the value entered by the user in the login form will be matched with the `phone_number` attribute;
  - length of the confirmation code;
  - validation time of the confirmation code;
  - number of attempts to enter the confirmation code for 1 login;
  - total number of attempts (number of code sends and code entry attempts, after which this authentication method will be temporarily blocked for the user);
  - blocking time when attempts are exceeded (in minutes);
  - configure how to send the code:
    - \* send push notification - you should specify an attribute with a cell phone number or other user ID required by the service, for example, `${phone_number}`;
    - \* send SMS - specify attribute with user's cell phone number, for example, `${phone_number}`;

Authentication by code sent via SMS/push

In order to correctly identify the user, specify how the username should be formed and which attribute in the data source it corresponds to. You can create a number of alternative rules to define the login. The login is not case-sensitive.

To create a rule, use [substitution strings](#). For example, the rule `CN=${login}` means the string entered by the user will be compared to the `CN` attribute in the data store.

[View substitution strings](#)

phone\_number = \$(login) ✖

+ add condition  
+ add an alternative rule

Confirmation code parameters

Length: 6  
Number of symbols in the one-time password

Duration: 300  
Timeout in seconds after which the one-time password is no longer valid. User needs to enter a new password

Number of attempts for 1 login: 50  
The number of failed attempts to enter the verification code in a single login attempt. If the number of attempts is exceeded, a new code must be sent.

Total attempts: 5  
The total number of verification code submissions and verification code attempts that will cause the authentication method to be temporarily blocked

Blocking time when attempts are exceeded, in min.: 3  
During the specified time, the authentication method will be unavailable to the user

How to send a code

Configure how to send the confirmation codes. If more than one method is selected, the first one will be considered as the primary method and the others as backups.

Send	Attribute with contact	
SMS	\$(phone_number-)	✖

+ Add a sending method

Cancel Save

– rule for selecting an attribute store to search for a phone number entered by a user.

Attribute store selection rules

By default, users are searched for authentication in all active attribute stores. In these block you can configure rules, in case of which the user will be searched only in specified attribute store.

Several alternate attribute store selection rules can be set. This will allow you to authenticate some users through one repository, and others through another one.

To create a rule, use [substitution strings](#).

[View substitution strings](#)

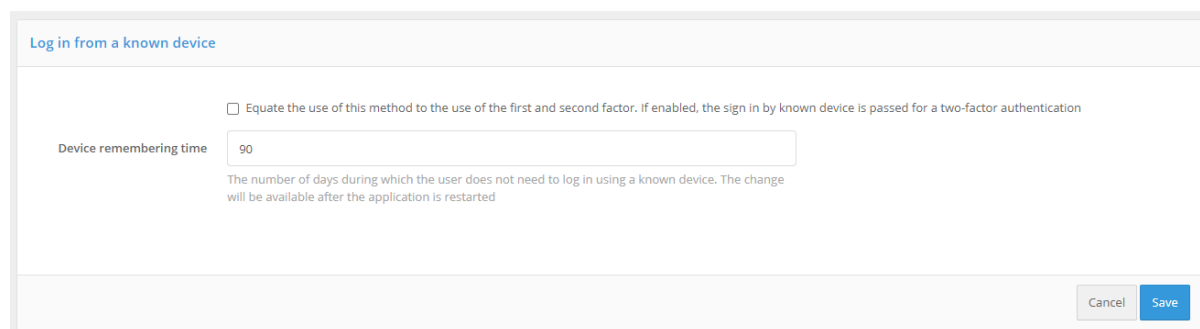
[Create rule](#)

- configure Blitz Identity Provider connection to [SMS gateway and the](#) (page 152) push notification service.

## Logging in from known device

Login from a known device allows for not requesting user identification and authentication (first factor method) if the user has, within a certain period of time, already logged in from that device and browser. In other words, the user can log in without authentication after restarting the browser.

Setting the method includes specifying the duration of device memorization. It can also be set to not require two-factor authentication when logging in from a memorized device (option “*Equate the use of this method to the use of the first and second factor*”). If this option is enabled, logging in from a known device will mean that the user has passed two-factor authentication.



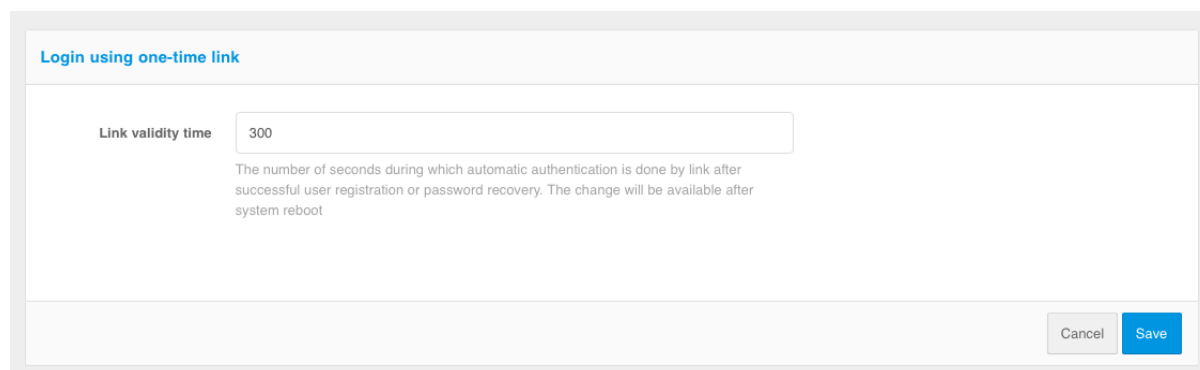
The screenshot shows a configuration window titled "Log in from a known device". At the top, there is a checkbox with the text: "Equate the use of this method to the use of the first and second factor. If enabled, the sign in by known device is passed for a two-factor authentication". Below this, there is a text input field labeled "Device remembering time" containing the value "90". A small explanatory text below the input field reads: "The number of days during which the user does not need to log in using a known device. The change will be available after the application is restarted". At the bottom right of the window, there are two buttons: "Cancel" and "Save".

## Logging in by one-time link

One-time link login is used to provide automatic login after a user has self-registered an account, recovered a forgotten password, or when using a special login mode when opening a web browser from a mobile application to which the user has previously logged in.

**Note:** [Learn more](#) (page 324) about the last use case.

Method customization includes specifying the validity time of the link used for automatic login. For automatic login to work, no more than the time specified in the setting must have elapsed from the time the link is generated (after successful completion of registration or password recovery or receipt of the `css` parameter by the mobile application) until the user login is initiated, and that the link has not been used before.



The screenshot shows a configuration window titled "Login using one-time link". It features a text input field labeled "Link validity time" with the value "300". Below the input field, a small explanatory text reads: "The number of seconds during which automatic authentication is done by link after successful user registration or password recovery. The change will be available after system reboot". At the bottom right of the window, there are two buttons: "Cancel" and "Save".

## Logging in by QR code

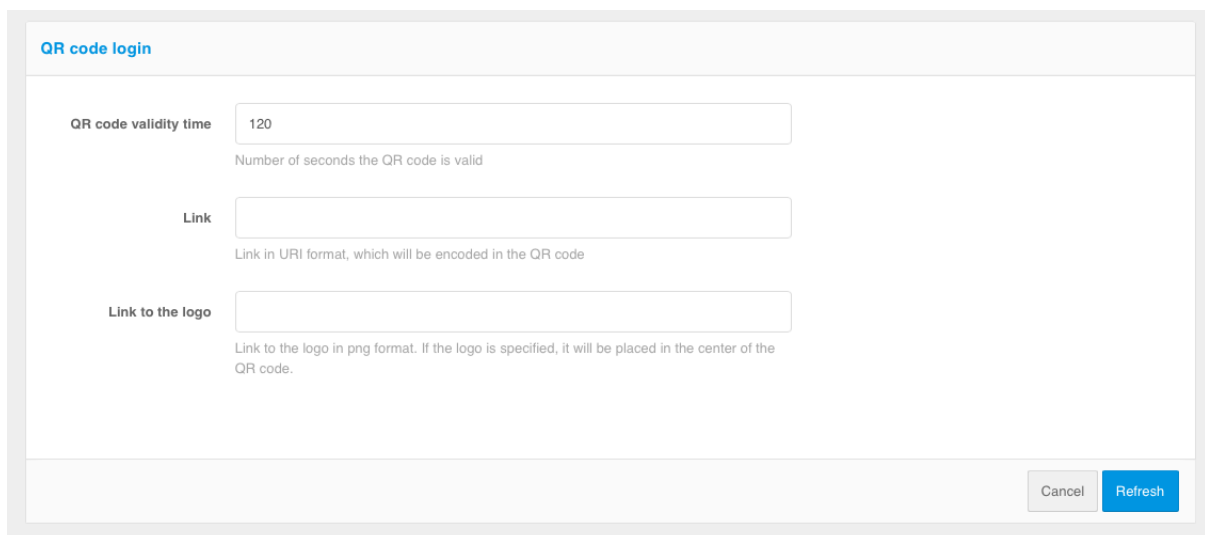
Blitz Identity Provider provides the option to set up a QR code login to the web application as the first authentication factor.

The login process is organized as follows:

- A user in a browser initiates a login to a web application. Blitz Identity Provider displays a login page. On the login page, the user selects “Login by QR Code”.
- Blitz Identity Provider displays a QR code and instructions to the user on the login page. The QR code has a limited validity period (a timer with the QR code validity period is shown to the user).
- The user launches the mobile application of the company, which has built-in support for the QR code login mode, and scans the QR code with the help of this application.
- The mobile app shows the user detailed login information received from Blitz Identity Provider (the name of the app being logged into, the IP address, browser, and operating system name of the device being logged into).
- The user in the mobile app decides whether to allow or deny entry.
- Depending on the user’s decision on the computer, the user successfully logs into the application or the login request is denied.

Customizing the method includes specifying the following parameters:

- QR code validity time - during this period the user must scan the QR code and make a decision to log in;
- link that will be encoded in the QR code - indicates which application or web page should be launched in case the QR code is read by the standard “Camera” application. The encoded QR code will be passed to the link as a parameter (the link will be `QR_URL?code=b0671081-cb73-4839-8bc1-8cf020457228`);
- logo link (optional) - this logo will be displayed in the center of the QR code.



The screenshot shows a configuration window titled "QR code login". It contains three input fields with labels and descriptions:

- QR code validity time**: A text input field containing the value "120". Below it, the text reads "Number of seconds the QR code is valid".
- Link**: A text input field. Below it, the text reads "Link in URI format, which will be encoded in the QR code".
- Link to the logo**: A text input field. Below it, the text reads "Link to the logo in png format. If the logo is specified, it will be placed in the center of the QR code."

At the bottom right of the form, there are two buttons: "Cancel" (disabled) and "Refresh" (active).

### Automatic user identification by session properties

Blitz Identity Provider can perform automatic user identification and grant access based on pre-calculated session properties. Any session properties that can be defined by the Customer's tools and provided in Blitz Identity Provider are supported.

**Tip:** A special case of using the method is the user logging in using a mobile phone number that is automatically determined by its IP address by the Customer-the mobile operator.

**Attention:** Automatic identification is possible only for the first factor.

To use this authentication method, follow the steps described below.

#### Step 1. Create the login procedure

To use automatic identification, you must *create* (page 191) a login procedure performed before passing the first authentication factor, which will request session properties from the Customer's service. For example, in a special case, when logging in using an automatically determined phone number, the procedure should perform the following actions:

1. Determining the user's IP address. If the IP address is in the specified range, the Customer's mobile operator service is called to determine the mobile phone number.
2. After receiving the phone number, the procedure requests Blitz Identity Provider to log in using the automatic identification method.

#### Step 2. Add a method to blitz.conf

In order for the automatic identification method to be displayed on the tab Authentication -> First factor, follow these steps:

1. Open the configuration file `/usr/share/identityblitz/blitz-config/blitz.conf`.

```
sudo vim /usr/share/identityblitz/blitz-config/blitz.conf
```

2. Add the method to the list of available methods of the first factor of the block `blitz.prod.local.idp.login.factors` by analogy with the example below. The methods of the first factor are specified in the first section of the block. The name of the method should consist of the prefix `sprop` and an identifier: for example, the method `sprop_msisdn` from the example has the identifier `msisdn`.

**Note:** You can add several methods.

```
"login" : {
  "factors" : [
    [
      {
        "enabled" : false,
        "method" : "sprop_msisdn"
      },
      ...
    ],
    [
      ...
    ]
  ]
}
```

(continues on next page)

(continued from previous page)

```

    ],
    ...
}

```

- Restart the services.

```
sudo systemctl restart blitz-idp blitz-console
```

### Step 3. Configure the method in the console

The configuration of the method in the admin console is performed as follows:

- In the admin console, go to Authentication -> First factor -> method settings Automatic identification.
- Map the attribute stored in the Blitz Identity Provider data source to the session property received from the Customer's service when performing the login procedure. After receiving the session property, Blitz Identity Provider will search for its value among the values of the specified attribute and, if successful, will allow logging in to the corresponding account. For example, mapping `phone_number=${p_msisdn}` means that the session property `p_msisdn` will be compared with the attribute `phone_number` in the data store.

**Tip:** You can add several search conditions among the attributes that must be fulfilled simultaneously in order for the user to be identified, as well as enter an alternative rule.

Automatic identification

Identifier

In order to correctly identify the user, specify how the username should be formed from session properties and which attribute in the data source it corresponds to. You can create several alternative rules.

For example, the rule `phone_number=${p_msisdn}` means that the session property `p_msisdn` will be compared to the `phone_number` attribute in the data store.

=

[+ add condition](#)

[+ add an alternative rule](#)

Do not show the user the login confirmation page

Display user id

The expression is formed from the current user attributes. The calculated result is displayed to the user in the login consent page.

- By default, after the user is automatically identified, their ID and a login confirmation request are displayed on their screen. Set a rule for generating a user ID from its attributes as a substitution string. This may be a disguised phone number, username, etc.

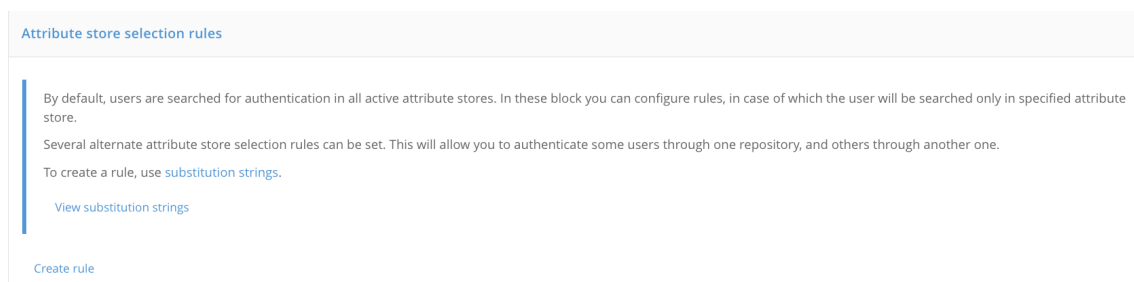
To deactivate the login confirmation, check the box Do not show the login confirmation screen to the user.

- Click Save.
- By default, users are searched for authentication in all active repositories. In the block Attribute Store Selection Rules you can set up rules that will search for a user in a specific store. You can set several alternative storage selection rules. This will allow you to authenticate some users using one storage, and others using another.

To create a rule, use the following components:

- `flag not`: indicates that the condition is inverted;
- the first column is the expression to be checked, for example, an attribute of an account, an application identifier, etc.;
- the second column: the selection condition in the form of a regular expression, for example, the value of the user attribute, the value of the application identifier, etc.

For example, in order to authenticate all users whose phone number contains the code 980 in the specified storage, create a rule as shown in the figure below.



6. Click Save.

#### Step 4. Customization of texts

If you use several methods of *automatic identification* (page 88), you should customize the interface texts for each of them, guided by *algorithm* (page 234).

You will need to include the method name or method identifier in the text string identifier. Method name *is defined* (page 88) in the configuration file `/usr/share/identityblitz/blitz-config/blitz.conf` and consists of the prefix `sprop_` and the method identifier: for example, the method `sprop_msisdn` has the identifier `msisdn`.

The following methods and strings are used for customization:

##### Login form

Customization using the method name `<sprop_id>`:

```
login.methods.sprop.head.title.<sprop_id>=Confirm log-in with phone number
login.methods.sprop.info.<sprop_id>=Your phone number<br><strong>{0}</strong>.
login.methods.sprop.btn.consent.<sprop_id>=Log in
login.methods.sprop.btn.refuse.<sprop_id>=Log in with another phone number
```

##### Displaying the method in the list of available methods during authentication

Customization using the method identifier `<id>`:

```
login.methods.switcher.title.sprop.<id>=Autologon with phone number
login.methods.switcher.label.sprop.<id>=Autologon with phone number
```

## Displaying a method in the list of methods in the admin console

Customization using the method name `<sprop_id>`:

```
page.authn.<sprop_id>.title=Autologon with phone number
page.authn.<sprop_id>.info=To identify a user, we analyze a session property p_
↳msisdn which is calculated and saved when the authentication flow starts.
```

## Form of method configuration in the console

Customization using the method name `<sprop_id>`:

```
page.method.sprop.title.<sprop_id>=Autologon with phone number
page.method.sprop.info.<sprop_id>=<p>In order to correctly identify the user,
↳specify how the username should be formed from session properties and which
↳attribute in the data source it corresponds to. You can create several
↳alternative rules. </p>For example, the rule <code>phone_number=${p_msisdn}'</
↳code> means that the session property <code>p_msisdn</code> will be compared to
↳the <code>phone_number</code> attribute in the data store.</p>
```

## The result of executing the method on the Events tab of the admin console

- Successful login: add the line `audit.method.<sprop_id>`.
- Login failed: add the line `console.audit.type.auth_failed.<sprop_id>`.

```
audit.method.<sprop_id>=Autologon with phone number
console.audit.type.auth_failed.sprop_msisdn=Error when logging in with phone number
```

## Displaying an unsuccessful login event in the User profile

To display an unsuccessful login in the User profile, add the line `profile.audit.type.auth_failed.<sprop_id>`.

```
profile.audit.type.auth_failed.<sprop_id>=Error when logging in with phone number
```

## Log-in confirmation with a HMAC-based one-time password (HOTP)

Any hardware key fob compatible with the [RFC4226](https://tools.ietf.org/html/rfc4226) “HOTP: An HMAC-Based One-Time Password Algorithm”<sup>25</sup> standard can be used to verify the second factor of authentication using the One-Time Secret-based Password (HOTP) authentication method.

To use HOTP, you must:

- configure and enable this authentication method;
- upload a HOTP device description file to Blitz Identity Provider. The description file is provided by the HOTP device provider. To upload the description file, use the “Devices” menu section in Blitz Identity Provider admin console;
- bind the HOTP device to the user account and issue the HOTP device to the user. Binding can be done in two ways - either the administrator binds the device by serial number to the user account in the Management Console under the “Users” menu, or the user binds the device to his/her account by himself/herself using the “My Account” web application.

<sup>25</sup> <https://tools.ietf.org/html/rfc4226>



**Hardware-generated one-time passwords (HOTP)**

Here you can specify general properties of the hardware-generated one-time passwords (HOTP) method. Specific properties are defined when a device is attached to a user account.

Permissible deviation   
The number of subsequent codes that can be entered for a successful login (look-ahead window)

Deviation for synchronization   
The range of codes within which the search is performed during the synchronization process

To configure the “One-time secret-based password (HOTP)” authentication method, you must set:

- maximum allowable deviation during code verification - the number of subsequent codes (for example, if the user accidentally pressed the button to generate a new password and did not use it during the authentication process) at which the authentication will be successful. If the user enters the correct code, Blitz Identity Provider will automatically resynchronize with the device;
- reject for synchronization - if the user repeatedly presses the code generation button on the device and does not use the code to confirm the login, the device will cease to be synchronized with the server. In this case, the next time the user logs into Blitz Identity Provider, he or she will be prompted on the login page to go through the device reconciliation procedure. To do this, the user will enter three confirmation codes sequentially generated by the device. Blitz Identity Provider will then check whether the code sequence entered by the user is encountered according to the “*Reject for synchronization*” setting and will resynchronize with the device if successful;
- total number of attempts - number of attempts to enter the confirmation code, after which this confirmation method will be blocked;
- blocking time when attempts are exceeded (in minutes).

### Time-based one-time password log-in confirmation (TOTP)

Any devices and programs compatible with the [RFC6238 “TOTP: Time-Based One-Time Password Algorithm”](https://tools.ietf.org/html/rfc6238)<sup>26</sup> standard may be used to verify the second factor of authentication using the Time-Based One-Time Password (TOTP) authentication method. These may include:

- hardware keyfobs (one-time password generators) based on time;
- mobile apps.

**Note:** The most well-known applications for generating TOTP codes are Google Authenticator, Twilio Authy, FreeOTP Authenticator, Microsoft Authenticator.

In the settings for the authentication method “*Time-based One-Time Password (TOTP)*”, you must specify:

1. Allowable code validation deviation (number of previous / next codes). By default, both values are 1: a user can enter both the current validation code and the next or previous one (generated in neighboring time intervals) when logging in. Such a need may arise, for example, to compensate for possible minor unsynchronization of server time and time on TOTP-devices of users.
2. Total number of attempts - number of attempts to enter the confirmation code, after which this confirmation method will be blocked.
3. Blocking time when attempts are exceeded (in minutes).

<sup>26</sup> <https://tools.ietf.org/html/rfc6238>

4. Customize the display of one-time password generators, which includes *“Attribute with user name”* and *“Name of the single sign-on system”*. These settings will be displayed in the mobile app after the user account is linked.
5. Links to one-time password generator applications. Links to applications that are recommended to be used by users should be specified. These links will be offered to the user in the web application User profile.

**Time-based one-time passwords (TOTP)**

Here you can specify general properties of the time-based one-time passwords (TOTP) method. Specific properties are defined when a mobile application is attached to a user account.

Permissible deviation (ahead)   
The number of subsequent codes that can be entered for a successful login (look-ahead window)

Permissible deviation (backward)   
The number of subsequent codes that can be entered for a successful login (look-backward window)

---

**Configuring the appearance of the one-time password generators**

Attribute with username   
The username will be displayed in the one-time password generator after binding

The name of the single sign-on system   
The name of the single sign-on system will be displayed in the one-time password generator after binding

---

**Links to one-time password generator mobile applications**

Specify for each operating system what mobile application you recommend to use to generate one-time passwords. If the link is not specified users won't see the recommendation for this operating system.

IOS

Android

Windows Mobile

## Binding devices to user accounts

Binding HOTP and TOTP devices via the Admin console differs depending on whether key fob hardware or mobile apps are used.

## Binding of hardware keyfobs

To be able to use hardware HOTP and TOTP devices as authentication tools, the administrator must first load a file with the device batch descriptions received from the device vendor in the *“Devices”* menu of the Admin Console. The file contains information about the device serial number, initialization vector, and a number of other settings. Blitz Identity Provider supports uploading of common file formats (specialized XML files, CSV files) of device description files from different device manufacturers.

**One-time password security tokens loading**

Load a file with generators' data. Once uploaded, users can bind HOTP/TOTP-generators to their accounts by entering the serial number.

Generator name

Data format

File with data

---

**Loaded security tokens**

To perform a file upload, you must specify a name for the uploaded generators (it can be, for example, the device name), the data format, and the path to the file with device descriptions. When you click the *“Download”* button, Blitz Identity Provider will report how many device records were loaded or discarded (if their description in the file was incorrect or the device record is already present in the system).

An example of a downloadable Aladdin/SafeNet XML format file for HOTP devices with the SHA-1 algorithm with a minimum set of parameters:

```
<?xml version="1.0" encoding="utf-8"?>
<Tokens>
  <Token serial="SN123">
    <Applications>
      <Application>
        <Seed>7bba106e428231c4d4e78361375d161c2d59b40b</Seed>
        <MovingFactor>0</MovingFactor>
      </Application>
    </Applications>
  </Token>
</Tokens>
```

Explanation of the parameter values in the file:

- `serial` - serial number of the device.
- `Seed` is the device key in hexadecimal (hex) format.

**Note:** If a software one-time code generator is used to emulate a HOTP device, a Base32 string is usually entered as a secret in the software generator. In this case, the value from `Seed` must be converted from hex to Base32, and the resulting value must be used in the program generator.

- `MovingFactor` - initial value of the generator (usually 0).

Under *“Devices”* you can also search for a device by serial number and see, if and to which account the found device has been bound.

After loading the file you should:

- go to the account of the user to whom you want to bind the device (menu *“Users”*, see [Binding devices for 2FA with a one-time password](#) (page 142));

- find the “*Time-based password generator (TOTP)*” or “*Secret-based password generator (HOTP)*” section;
- select “*Another type*»;
- enter the serial number of the required device and the current one-time confirmation code.

Time-based one-time password generator (TOTP)

Serial number

Serial number of the device for generating one-time passwords

Value

### Binding a mobile application

To bind a mobile application you must:

- go to the account of the user to whom you want to bind the mobile application (menu “*Users*”, see [Binding devices for 2FA with a one-time password](#) (page 142));
- find the section “*Time-based password generator (TOTP)*”;
- select «*GoogleAuthenticator*»;
- edit the name of the mobile application, if necessary;
- using the mobile application, take a picture of the displayed QR code or enter a secret line into the application.

The user can also independently link the mobile application generating TOTP codes in the web application “*User profile*”.

Time-based one-time password generator (TOTP)


Application name

Encryption algorithm

Password length   
The number of symbols in a one-time password

Password refresh time   
A new password will be generated when the time (in seconds) expires

Secret   
Secret is Base32 encoded



### Confirmation codes sent in SMS and push notifications

You can use push notifications sent to the mobile app or SMS messages for login confirmation (the second authentication factor).

To use the confirmation codes, you must:

- configure and enable the authentication method Confirmation via SMS/push. For the method to work correctly, it is necessary to define:
  - length of the confirmation code;
  - validation time;
  - number of attempts to enter the confirmation code for 1 login;
  - total number of attempts (number of code sends and code entry attempts, after which this authentication method will be temporarily blocked for the user);
  - blocking time when attempts are exceeded (in minutes);
  - configure the sending methods:

- \* send push notification - you should specify an attribute with a cell phone number or other user ID required by the service, for example, `${phone_number}`;
- \* send SMS - specify attribute with user's cell phone number, for example, `${phone_number}`;
- configure Blitz Identity Provider connection to the SMS gateway and push notification service (see [Notifications and sending messages](#) (page 152)).

**Attention:** If the user does not have a mobile phone number, he will not be able to use method of login verification by confirmation code sent via SMS.

#### Confirmation by code sent via SMS/push

##### Confirmation code parameters

Length	<input type="text" value="6"/>	Number of symbols in the one-time password
Duration	<input type="text" value="300"/>	Timeout in seconds after which the one-time password is no longer valid. User needs to enter a new password
Number of attempts for 1 login	<input type="text" value="3"/>	The number of failed attempts to enter the verification code in a single login attempt. If the number of attempts is exceeded, a new code must be sent.
Total attempts	<input type="text" value="5"/>	The total number of verification code submissions and verification code attempts that will cause the authentication method to be temporarily blocked
Blocking time when attempts are exceeded, in min.	<input type="text" value="3"/>	During the specified time, the authentication method will be unavailable to the user

##### How to send a code

Configure how to send the confirmation codes. If more than one method is selected, the first one will be considered as the primary method and the others as backups.

Send	Attribute with contact	
<input style="width: 90%;" type="text" value="SMS"/>	<input style="width: 90%;" type="text" value="\${phone_number-}"/>	<span style="color: red; font-weight: bold;">✕</span>
<a href="#">+ Add a sending method</a>		

## Confirmation codes sent by email

You can use emailed confirmation codes to confirm the log-in.

**Confirmation by email**

---

**Confirmation code parameters**

Length   
Confirmation code character limit

Expiration time   
The number of seconds before the confirmation code will be invalid. Sending a new code is required

Number of attempts for 1 login   
The number of failed attempts to enter the verification code in a single login attempt. If the number of attempts is exceeded, a new code must be sent.

Total attempts   
The total number of verification code submissions and verification code attempts that will cause the authentication method to be temporarily blocked

Blocking time when attempts are exceeded, in min.   
During the specified time, the authentication method will be unavailable to the user

---

**Sending options**

Attribute with contact   
Expression that will generate the email address to send the confirmation code

To do this, you must:

- configure and enable this authentication method. The method must be defined for it to work correctly:
  - length of the confirmation code;
  - validation time;
  - number of attempts per log-in to enter the confirmation code;
  - total number of attempts (number of code sends and code entry attempts, after which this authentication method will be temporarily blocked for the user);
  - blocking time when attempts are exceeded (in minutes);
  - configure the sending method: specify the attribute in which the user's e-mail address is stored, e.g. `{email}`;
- [configure](#) (page 152) Blitz Identity Provider connection to the SMTP service.

## Log-in confirmation via Duo Mobile

You can use the [Duo Mobile app](#)<sup>27</sup> (a Cisco company) to confirm the login (the second factor of authentication).

To do this, you need to make adjustments on the Duo Security service side:

- register an account on the [Duo website](#)<sup>28</sup>;
- log in to the [administrator panel](#)<sup>29</sup> and go to the Applications section;
- click on Protect an Application, among the applications find `Auth API`. Then click on Protect this Application to get your integration key, secret key and hostname.

Once these operations are complete, you need to make settings in Blitz Identity Provider Admin Console.

- configure the authentication method Duo push-authentication. You must specify:
  - Duo account parameters (host name, integration and secret keys);
  - interaction properties:
    - \* user name pattern (set in the substitution string) - this name will be displayed in Duo Mobile as the account name;
    - \* enrollment code validity time (in seconds) - the time the enrollment code will be valid for QR code;
  - data to be displayed in the application - information displayed to the user in Duo Mobile in the form of "key: value". Here you can pass a custom attribute value or some fixed value. You can also specify the string `${app}` as a value - this will display the name of the application the user is logged into;
  - links to application - Duo Mobile.
- enable the Duo push-authentications method in the Authentication section.

<sup>27</sup> <https://duo.com/product/multi-factor-authentication-mfa/duo-mobile-app>

<sup>28</sup> <https://signup.duo.com/>

<sup>29</sup> <https://admin.duosecurity.com/>



### Duo push-authentication properties

To use Duo Security push-authentication you should:

- register a [Duo account](#);
- enter [Duo admin panel](#) and go to Applications;
- press Protect an Application, find Auth API among the applications. Then press Protect this Application to get your hostname, integration and secret keys.

---

#### Account

API hostname

Integration key

Secret key

---

#### Interaction properties

Username pattern   
The substitution string that defines the username in the application that accepts push notifications. For example: "\${mail}"

Enrollment code validity time   
The number of seconds the enrollment code should be valid for QR-code

---

#### Data to be displayed in the application

You can show in the mobile application some information as "key: value" data. Set the required keys and their values using the substitution strings. For example, `Name = ${name}` `$(surname)` will display the key `Name` with values from `name` and `surname` attributes.

[See substitution strings](#)

Имя пользователя	=	\${name}	<span style="color: red; font-weight: bold;">✖</span>
------------------	---	----------	---

[+ Add](#)

---

#### Links to application - Duo Mobile

Specify for each OS which mobile applications are recommended to download for push authentication. If the link is not specified, users will not be prompted to download the application for this OS.

iOS	<input type="text" value="https://itunes.apple.com/ru/app/duo-mobile/id422663827"/>
Android	<input type="text" value="https://play.google.com/store/apps/details?id=com.duosecurity.duomobile"/>
Windows Mobile	<input type="text" value="https://www.microsoft.com/ru-ru/store/p/duo-mobile/9nblggh08m1g"/>

You can bind the Duo Mobile app to your user account in the following ways:

- by the user independently through the web application User profile;
- by an administrator through the Admin console.

In the web application User profile the user should go to the section Security / Login Confirmation and perform the following steps:

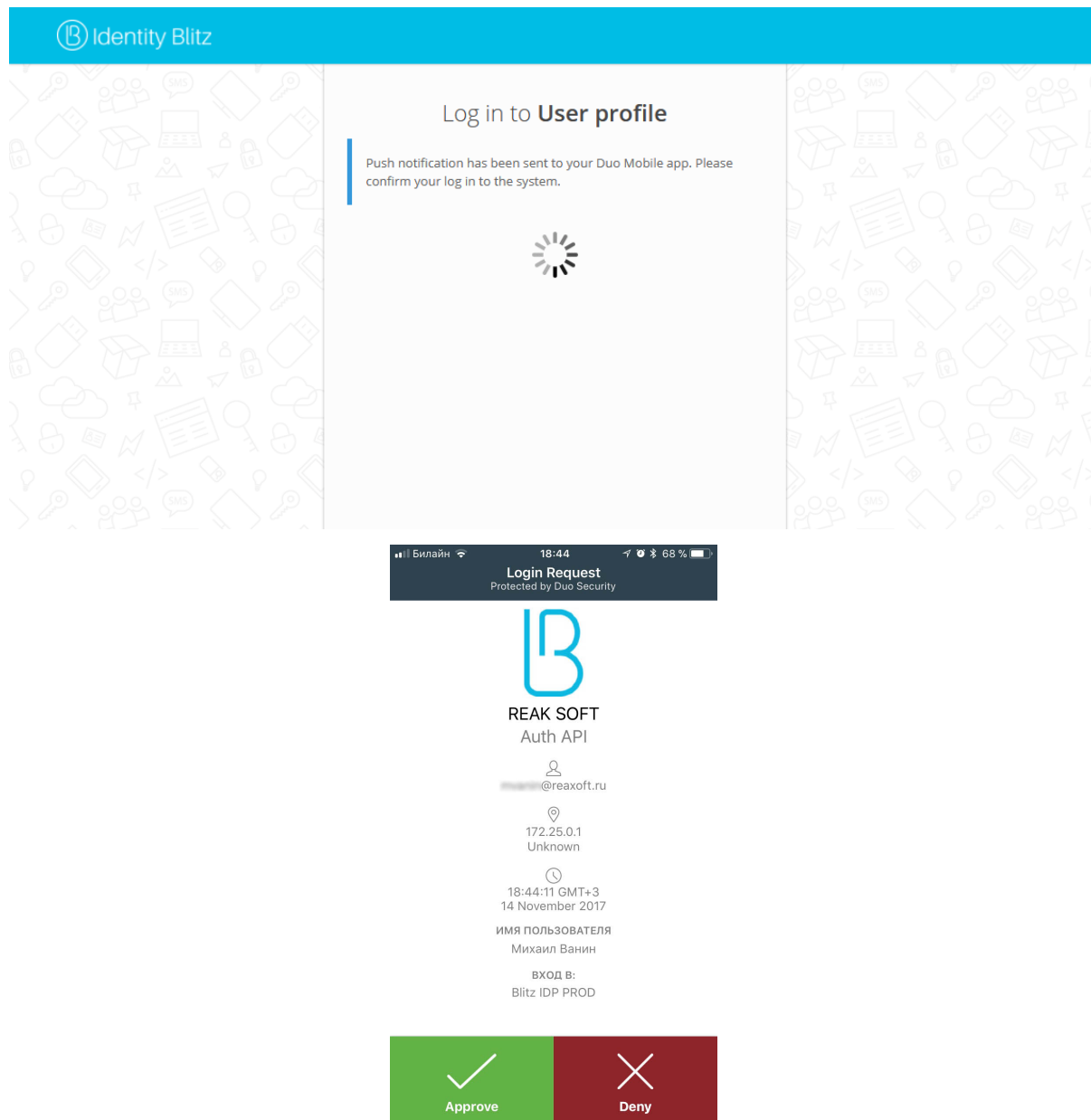
1. Select the login confirmation method - Confirmation via mobile application Duo Mobile.
2. Install the Duo Mobile app on your smartphone and scan the QR code and press Confirm.

3. After verification, this authentication method will be added to the user.

In the admin console, the administrator must:

1. Find the user required.
2. Go to the Application Duo Mobile (QR Code) box and click on the Link Duo Mobile button.
3. Ask user to scan the QR-code with the Duo Mobile application.

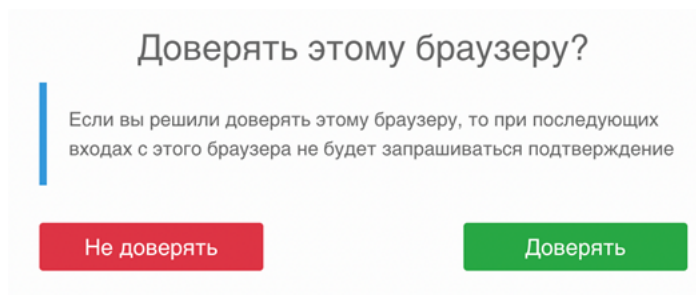
The pictures show an example of the login page appearance when confirming entry using push-notification in the Duo Mobile application.



### Re-confirmation when logging in from known device

Blitz Identity Provider remembers the devices on which a user has confirmed login during the login process using one of the login confirmation methods supported by Blitz Identity Provider.

You can configure the login procedure to display a screen asking if the user trusts the browser after a successful login confirmation, so that repeated logins from this device and browser do not prompt the user for login confirmation.



If the user logs in again from a trusted browser, the user will not be asked for login confirmation if `bdg-primary:Input from a known device` authentication method is enabled in the `bdg-primary:Authentication` menu in the `bdg-primary:Second factor` block.

### Confirmation by answering security question

Blitz Identity Provider allows you to request the user to enter the answer to the security question to confirm the login. This can be useful in confirmation scenarios when recovering a forgotten password. To use this authentication method, follow the steps described below.

#### Step 1. Add method to blitz.conf

In order for the authentication method Confirmation by the answer to the security question to appear in authentication methods on the tab Second factor, follow these steps:

1. Open the `/usr/share/identityblitz/blitz-config/blitz.conf` file.

```
sudo vim /usr/share/identityblitz/blitz-config/blitz.conf
```

2. In the settings section `blitz.prod.local.idp.login.factors` in the second list, add a block of settings using the `secQsn` method:

```
"login" : {
  "factors" : [
    [
      ...
    ],
    [
      {
        "enabled" : false,
        "method" : "secQsn"
      },
      ...
    ]
  ],
  ...
}
```

3. Restart the services.

```
sudo systemctl restart blitz-idp blitz-console blitz-recovery
```

## Step 2. Create directory of security questions

To create a directory of security questions, follow these steps:

1. Create the directory `/etc/blitz-config/custom_messages/dics` on the server.
2. Create a file `/etc/blitz-config/custom_messages/dics/securityQuestions` with the contents of the checklist. Example of a `securityQuestions` file with a directory of security questions:

```
01=What is your mother's maiden name?
02=What is your grandmother's maiden name?
03=What was the first movie you saw in the cinema?
04=What is your favorite literary work?
05=What was the name of your third grade teacher
06=The first dish you learned to cook
07=What was the name of your first pet
08=What did you want to become as a child?
09=What was the name of the first school you went to?
10=What was the name of the first street where you lived as a child?
```

**Attention:** The number in the checklist is used for sorting when displaying a list of security questions to the user.

3. Check the owner of the `dics` directory and the directory files in it. The owner must be `blitz:blitz`.

```
chown -R blitz:blitz /etc/blitz-config/custom_messages/dics
```

4. In the configuration file `/usr/share/identityblitz/blitz-config/blitz.conf`, add the `"dics"` block to the `blitz.prod.local.idp.messages` block. In the `names` setting, specify the name of the `securityQuestions` directory. For example:

```
"dics" : {
  "dir" : "custom_messages/dics",
  "names" : [
    "securityQuestions"
  ]
}
```

## Step 3. Configure method in console

The following settings must be set in the Admin console:

- Total number of attempts – the number of attempts to enter the answer to the security question, after which this confirmation method will be blocked.
- Blocking time when attempts are exceeded (in minutes).

The list [configured](#) (page 103) of security questions is also displayed in the admin console.

### Confirmation by the answer to the security question

Here you can specify basic settings of the method. The list of security questions is read-only. To edit this list, you need to make corrections to the file with the custom messages.

**Total attempts**

The total number of answer input attempts that will cause the authentication method to be temporarily blocked

**Blocking time when attempts are exceeded, in min.**

During the specified time, the authentication method will be unavailable to the user

---

**List of available secret questions**

### Confirmation by incoming call

Blitz Identity Provider allows you to transfer one-time codes to implement the second authentication factor in the incoming call number (Flash Call method). In this case, after successful initial authentication, a call will be made to the user's number from a previously unknown phone number, the last digits of which will need to be entered to confirm login. The call is made with the user's permission.

To configure the Flash Call method, follow the steps described below.

#### Step 1. Add the method to blitz.conf

In order for the authentication method Confirmation by Incoming call to appear in authentication methods on the tab Second factor, follow these steps:

1. Open the `/usr/share/identityblitz/blitz-config/blitz.conf` file.

```
sudo vim /usr/share/identityblitz/blitz-config/blitz.conf
```

2. In the settings section `blitz.prod.local.idp.login.factors` in the second list, add a block of settings using the `flashCall` method:

```
"login" : {
  "factors" : [
    [
      ...
    ],
    [
      {
        "enabled" : false,
        "method" : "flashCall"
      },
      ...
    ]
  ],
}
```

(continues on next page)

(continued from previous page)

```
} ...  
}
```

### 3. Restart the services.

```
sudo systemctl restart blitz-idp blitz-console blitz-recovery
```

## Step 2. Configure the method in the console

In the Admin Console, follow these steps:

### 1. On the tab Confirmation by a Phone Call set the following settings:

- `Code length`: The number of last digits of the incoming call number to be used as a code on the second authentication factor.
- `Validity period`: The number of seconds after which the confirmation code ceases to be valid and a second call is required.
- `Number of attempts per login`: the number of failed attempts to enter the confirmation code during one login attempt. If the number of attempts is exceeded, a second call is required.
- `Total number of attempts`: the total number of confirmation codes sent and attempts to enter a confirmation code, after which this authentication method will be temporarily blocked.
- `Blocking time when the total number of attempts is exceeded, in minutes`: during the specified time, this authentication method will be unavailable to the user.
- `Name of the attribute with the user's mobile number`: Select from the list the attribute that stores the user's phone number for making a call.

To confirm authentication, a call is made from an unknown phone number, and user must enter the last digits of the incoming number

Code length	<input type="text" value="6"/>	Number of last digits of the incoming phone number
Expiration time	<input type="text" value="300"/>	Number of seconds after which the confirmation code no longer works and a new call is required
Number of attempts per log in	<input type="text" value="3"/>	Number of failed confirmation code attempts per login. If the number of attempts is exceeded, a new call is required
Total number of confirmation code attempts	<input type="text" value="5"/>	Total number of confirmation code attempts before the authentication method is temporarily locked
Lock period in minutes when the total number of code attempts is exceeded	<input type="text" value="3"/>	The authentication method will not be available to the user for the specified period
Total number of call attempts	<input type="text" value="10"/>	Total number of calls before the authentication method is temporarily locked
Lock period in minutes when the total number of call attempts is exceeded	<input type="text" value="5"/>	The authentication method will not be available to the user for the specified period
Attribute name with the users mobile phone number	<input type="text" value="phone_number"/>	A call is made to the phone from this user attribute

Click Save. As a result, the configuration of the method will be updated and the tab Phone Call Provider Driver will be displayed.

```

1 package flashcall;
2
3 import org.slf4j.LoggerFactory;
4 import org.slf4j.Logger;
5 import com.identityblitz.core.loop.http.HttpLoop;
6 import com.identityblitz.core.loop.http.HttpLoopRequest;
7 import com.identityblitz.core.loop.http.HttpLoopResult;
8 import com.identityblitz.core.loop.*;
9 import com.identityblitz.core.loop.http.*;
10 import com.identityblitz.json.JsonObj;
11 import java.util.Collections;
12
13 public class FlashCallFlow implements HttpLoop {
14     private final org.slf4j.Logger logger = LoggerFactory.getLogger("com.identityblitz.idp.flow.dynamic");
15
16     @Override
17     public HttpLoopRequest run(final JsonObj obj, final HttpLoopResult result) {
18         if (result == null) {
19             final String number = obj.asString("phone_number");
20             logger.trace("### flash call to = {}", number);
21             return HttpLoop.callBuilder("POST", "http://test.flashcall.ru/api/v1")
22                 .withHeader("X-Token", "1234567890")
23                 .withBody(JsonObj.empty.addString("id", "test_project").addString("dst_number", number.substring(number.length() - 10)))
24                 .withTimeout(20000)
25                 .build(JsonObj.empty);
26         } else if (result.status() == 200) {
27             final JsonObj body = result.body();
28             String callerInfo = body.asString("CallerID").substring(0, body.asString("CallerID").length() - 4) + "****";
29             return HttpLoop.ok(JsonObj.empty.addString("code", body.asString("CallerID")).addString("caller_info", callerInfo));
30         } else if (result.status() == 502) {
31             return HttpLoop.error("bad_gateway",
32                 Collections.<String, String>singletonMap("status", "" + result.status()));
33         } else {
34             return HttpLoop.error("wrong_http_status",
35                 Collections.<String, String>singletonMap("status", "" + result.status()));
36         }
37     }
38 }
39 }

```

Cancel Save

2. On the tab Phone Call Provider Driver set a Java procedure for integration with the REST service of the provider providing the dialer service, similar to the example below. To write the procedure, use the provider's documentation and the settings received during registration in the provider's service.

Listing 5: Example of a procedure for integration with a Flash Call provider

```
package flashcall;

import org.slf4j.LoggerFactory;
import org.slf4j.Logger;
import com.identityblitz.core.loop.http.HttpLoop;
import com.identityblitz.core.loop.http.HttpLoopRequest;
import com.identityblitz.core.loop.http.HttpLoopResult;
import com.identityblitz.core.loop.*;
import com.identityblitz.core.loop.http.*;
import com.identityblitz.json.JObj;
import java.util.Collections;

public class FlashCallFlow implements HttpLoop {
    private final org.slf4j.Logger logger = LoggerFactory.
↳getLogger("com.identityblitz.idp.flow.dynamic");

    @Override
    public HttpLoopRequest run(final JObj obj, final HttpLoopResult↳
↳result) {
        if (result == null) {
            final String number = obj.asString("phone_number");
            logger.trace("### flash call to = {}", number);
            return HttpLoop.callBuilder("POST", "http://
↳test.flashcall.ru/api/v1")
                .withHeader("X-Token", "1234567890")
                .withBody(JObj.empty.addString("id",
↳"test_project").addString("dst_number", number.substring(number.length() -↳
↳10)))
                .withTimeout(20000)
                .build(JObj.empty);
        } else if (result.status() == 200) {
            final JObj body = result.body();
            String callerInfo = body.asString("CallerID").
↳substring(0, body.asString("CallerID").length() - 4) + "****";
            return HttpLoop.Ok(JObj.empty.addString("code", body.
↳asString("CallerID")).addString("caller_info", callerInfo));
        } else if (result.status() == 502) {
            return HttpLoop.error("bad_gateway",
                Collections.<String, String>
↳singletonMap("status", "" + result.status()));
        } else {
            return HttpLoop.error("wrong_http_status",
                Collections.<String, String>
↳singletonMap("status", "" + result.status()));
        }
    }
}
```

**Tip:** [Learn more](#) (page 213) about custom errors implementation.

3. Enable the method Confirmation by Incoming Call in the list of methods on the tab Authentication -> Second factor.



## Configuring an external authentication method

Blitz Identity Provider, allows developers to add support for their own authentication method at deployment. To do this, you need to develop an application that implements the authentication logic and connect this application to Blitz Identity Provider. In Blitz Identity Provider, the authentication method “*External authentication method*” is configured for this purpose. You can implement an external authentication method to work as both a first and a second authentication factor.

**Adding an external authentication method**

**Identifier**   
Unique name (identifier) of the external authentication method. Will also be used in the audit

**Service URL**   
Address of the main service of the external authentication method. Takes current information about the authentication process as input and returns an HTTP-response, displayed to the user

**Names of assertions**   
Names of assertions, that the external method can set to the user

**Sent cookie**   
Cookies names which will be passed when calling the service method

**Sent headers**   
Headers names which will be passed when calling the service method

**URL of the applicability verification service**   
The address of the optional method service. If specified, this URL will be called before the invoking of the main service to check the applicability of the authentication method. If no URL is specified, it is assumed that the method is always applicable

**Security cookie**   
The name of the cookie, in which session identifier from the external method will be sent

**Sent assertions**   
List of assertions which will be sent to the external authentication method. If no list is specified, all available assertions are sent

**Additional parameters**   
Specify additional parameters that should be passed in the request to the external authentication method as json

To configure the use of Blitz Identity Provider with an external authentication method:

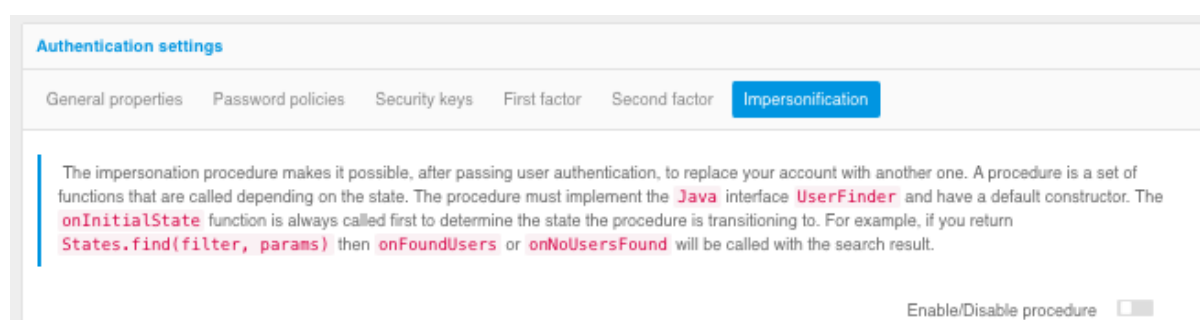
1. Configure a new “external” first or second factor authentication method by clicking the “*Add an external authentication method*” link. Specify the parameters of this authentication method:
  - method identifier - a card with the name of the method will be displayed among methods of authentication, the method with the given identifier will be possible to access from the Authentication flows;
  - URL of the external service;
  - assertion names - a list of assertions that an external method can set for the user;
  - passed cookies - list of names of cookies that will be thrown when an external method is called;

- sent headers - the list of headers, which will be passed when calling the external method;
  - Applicability Determination Service URL - address of the optional method service. If specified, this URL will be called before the main service is called to determine the applicability of this authentication method. If the URL is not specified, the method is assumed to be always applicable;
  - `cookie security` - the name of the `cookie` in which the session ID from the external method will be passed.
  - passed assertions - list of assertions to be passed to the external method (if the parameter is not specified, all assertions available in the login session will be passed to the external method);
  - additional parameters - specified in JSON format. The specified parameters will be passed to the external method. This can be useful to be able to configure the settings of the external authentication method through the Blitz Identity Provider admin console.
  - after saving enable method - a checkbox indicating that you should immediately enable the authentication method after saving the settings.
2. On the side of the external method it is necessary to provide the processing of authentication requests and check applicability according to the *Integration Guide* (page 294) document.

### Customizing the Impersonalization Procedure

Blitz Identity Provider allows you to customize the login process so that after the primary account has been authenticated and identified, the user can be prompted to select one of his secondary accounts for login.

The process of selecting auxiliary accounts is configured on the *“Impersonalization”* tab. For this purpose, an impersonation procedure is developed in Java. The text of the impersonation procedure can be saved, and after successful compilation, the procedure can be enabled using the *“Enable/disable procedure”* switch.



### 2.2.3 External identity providers



This section is dedicated to configuring login through external identity providers.

#### How to set up login via external identity providers

Setting up login via external identity providers has the following steps:

1. Make the settings in the section Identity providers in the Blitz Identity Provider admin console (see the sections in this section).
2. Perform settings on the Identity Provider side.
3. *Enable* (page 73) the ability to log in through this identity provider in the section Authentication.

The initial screen of the section Identity Providers shows the configured providers and allows you to select the required type of identity provider to configure.

Connected external identity providers		
Provider name	Unique name	Provider type
Google	google_1	google 
Facebook	facebook_1	facebook 

[Add provider](#)
[Google](#)
[Yandex](#)
[Mail ID](#)
[Facebook](#)
[VK](#)
[Odnoklassniki](#)
[ESIA](#)
[Sber ID](#)
[Blitz Identity Provider](#)

Configuring an Identity Provider consists of the following steps:

1. Specifies the vendor ID and vendor name.
2. Specifying the connection settings to the provider (described separately for each of the identity providers).
3. [Specifying the settings for linking the account](#) (page 121) of the external identity provider and the Blitz Identity Provider account. These settings are not provider-type specific.





## International providers

### Apple ID

To configure logging in via Apple ID, go to “Apple Developer Account” (note, the company must have a valid Apple Developer ID subscription) to the “[Certificates, Identifiers & Profiles](#)”<sup>30</sup> section, where you perform the following operations:

1. In the “*Certificates, Identifiers & Profiles*” window, select the “*App IDs*” filter in the upper right corner. Use the “+” button next to “*Identifiers*” to create a new “*App ID*”:
  - select the `App` type;
  - set “*Description*”. It will be displayed to the user in the Apple ID login confirmation window;
  - in “*Bundle ID*” set an identifier of the form `com.company.login` based on the domain used in Blitz Identity Provider;
  - in “*Capabilities*” check “*Sign In with Apple*”, click the `Edit` button next to it, and check that “*Enable as a primary App ID*” is selected;
  - you will be prompted to complete the configuration - all this is described in the following paragraphs. For now you should press “*Register*”.

## Certificates, Identifiers & Profiles

Certificates	Identifiers 	 App IDs 
Identifiers	NAME 	IDENTIFIER
Devices	BlitzIdentityProvider	com.identityblitz.blitzidp
Profiles		
Keys		
More		

<sup>30</sup> <https://developer.apple.com/account/resources/identifiers/list>

## Certificates, Identifiers & Profiles

[← All Identifiers](#)

### Register an App ID [Back](#) [Continue](#)



Platform  
iOS, macOS, tvOS, watchOS

Description  
  
You cannot use special characters such as @, &, \*, !, ", -, .

App ID Prefix  
Y2B5234KDQ (Team ID)

Bundle ID  Explicit  Wildcard  
  
We recommend using a reverse-domain name style string (i.e., com.domainname.appname). It cannot contain an asterisk (\*).

Capabilities App Services

ENABLED	NAME
<input type="checkbox"/>	 Access WiFi Information ⓘ
<input type="checkbox"/>	 App Attest ⓘ

- In the “Certificates, Identifiers & Profiles” window, select the “Services App IDs” filter in the upper right corner. Use the “+” button next to “Identifiers” to create a new “Services App ID”:

- set “Description”. It will be displayed to the user in the Apple ID login confirmation window;
- set “Identifier”. It is recommended to set it as `com.company.login` based on the domain used in Blitz Identity Provider. Later the created `Identifier` must be entered in Blitz Identity Provider settings as `client_id` in the “Service ID” setting;
- press “Register”;
- select the created “Service ID”. In its settings, check the “ENABLED” checkbox and click “Configure”;
- in the opened window, check that the “Primary App ID” contains the previously created “App ID”;
- in “Domains and Subdomains” list the domains used by Blitz Identity Provider, separated by commas;
- in “Return URLs” list the URLs of the return, separated by commas and specifying `https`. You must specify URLs, samples of which Blitz Identity Provider shows in the Apple ID connection settings, for example:

```
https://login.company.com/blitz/login/externalIdps/callback/apple/apple_1/false
https://login.company.com/blitz/profile/social/externalIdps/callbackPopup/
→apple/apple_1
```

- confirm the settings by pressing “Confirm”, “Done”, “Continue”, “Save” in successive screens;
- In the “Keys” menu, create a key for “Sign In with Apple”. This can only be done once, so it is recommended to save the created key somewhere. In Blitz Identity Provider this key is not currently used and will not be needed, but it should be created and saved for the future.

## Certificates, Identifiers & Profiles

[← All Identifiers](#)

### Register a Services ID [Back](#) [Continue](#)

Description  
  
You cannot use special characters such as @, &, \*, !, ", -, .

Identifier  
  
We recommend using a reverse-domain name style string (i.e., com.domainname.appname). It cannot contain an asterisk (\*).

# Certificates, Identifiers & Profiles

[< All Identifiers](#)

## Edit your Services ID Configuration

[Remove](#) [Continue](#)

Description:  Identifier: com.identityblitz.blitzidp-services

You cannot use special characters such as @, &, \*, +, ", ' -, .

ENABLED	NAME	
<input checked="" type="checkbox"/>	Sign In with Apple	<a href="#">Configure</a>

## Web Authentication Configuration

Use Sign in with Apple to let your users sign in to your app's accompanying website with their Apple ID. To configure web authentication, group your website with the existing primary App ID that's enabled for Sign in with Apple.

Primary App ID 1 App ID

BlitzIdentityProvider (Y2B5234KDQ.com.identityblitz.blit... ✕ ▾

### Website URLs +

Provide your web domain and return URLs that will support Sign in with Apple. Your website must support TLS 1.2 or higher. All Return URLs must be registered with the https:// protocol included in the URI string. After registering new website URLs, confirm the list you'd like to add to this Services ID and click Done. To complete the process, click Continue, then click Save.

Search ▾

Domains and Subdomains

demo.identityblitz.com 📄

Return URLs

https://demo.identityblitz.com/blitz/login/external... 📄

https://demo.identityblitz.com/blitz/profile/social/e... 📄

CancelDone

## Certificates, Identifiers & Profiles

< All Keys

### Register a New Key

Continue

Key Name

You cannot use special characters such as @, &, \*, ' ; " , - , .

ENABLE	NAME	DESCRIPTION	
<input type="checkbox"/>	Apple Push Notifications service (APNs)	Establish connectivity between your notification server and the Apple Push Notification service. One key is used for all of your apps. <a href="#">Learn more</a>	
<input type="checkbox"/>	DeviceCheck	Access the DeviceCheck and AppAttest APIs to get data that your associated server can use in its business logic to protect your business while maintaining user privacy. <a href="#">Learn more</a>	
<input type="checkbox"/>	MapKit JS	Use Apple Maps on your websites. Show a map, display search results, provide directions, and more. <a href="#">Learn more</a> ⓘ There are no identifiers available that can be associated with the key	Configure
<input type="checkbox"/>	Media Services (MusicKit, ShazamKit)	Access the Apple Music catalog and make personalized requests for authorized users, and check audio signatures against the Shazam music catalog. ⓘ There are no identifiers available that can be associated with the key	Configure
<input checked="" type="checkbox"/>	Sign in with Apple	Enable your apps to allow users to authenticate in your application with their Apple ID. Configuration is required to enable this feature. ⓘ This service must have one identifier configured.	Configure
<input type="checkbox"/>	ClassKit Catalog	Publish all of your ClassKit app activities to teachers creating Handouts in Apple Schoolwork. <a href="#">Learn more</a>	

Once you have completed the settings in the Apple Developer Account, you need to:

1. Go to the Blitz Identity Provider Admin console and add a provider that is of `Apple` type.
2. Fill in the Identity Provider settings:
  - Provider identifier;
  - Provider name;
  - The client identifier (`Service ID`) obtained in the Apple Developer Account.
3. Customize binding rules.
4. In the *"Authentication"* section of the Management Console, enable the use of the Apple Identity Provider authentication method.

**Apple basic properties**

Identity provider identifier   
Identity provider unique identifier. Is used only within Blitz Identity Provider

Identity provider name   
Identity provider display name. Is used only within Blitz Identity Provider

**Apple identity provider properties**

**Security**

Use the options from your [Apple Developer Account](#) to fill in the fields below. Do not forget to save the the Return URLs in the Domains section.

Return URLs   
  
These links should be entered in the identity provider settings in order to process the user authentication results. Use the https scheme if you are using a secure connection.

Service ID

Cancel Delete Save

## Google

To configure a Google login, you need to follow the steps below:

- Go to [Google API Manager](#)<sup>31</sup>, in which you perform the following operations:
  - go to “*Credentials*”;
  - create a project and create new credentials of the “*OAuth Client ID*” type;
  - type of new client identifier (e.g., web application) and give it a name;
  - do not set any restrictions, they will be specified later;
  - Google will generate the client ID and secret, it will be needed later in the Blitz Identity Provider admin console.
  - in “*Allowed redirect URIs*” list the URL return comma separated with `https`. You must specify URLs, samples of which Blitz Identity Provider shows in the Google connection settings, e.g.:

```
https://login.company.com/blitz/login/externalIdps/callback/google/google_1/
↪false
https://login.company.com/blitz/profile/social/externalIdps/callbackPopup/
↪google/google_1
```

The screenshot shows the Google Cloud Platform console interface for creating an OAuth client ID. The top navigation bar includes 'Google Cloud Platform', 'My Project 75123', and a search bar. The left sidebar is titled 'APIs & Services' and contains a menu with 'Dashboard', 'Library', 'Credentials' (selected), 'OAuth consent screen', 'Domain verification', and 'Page usage agreements'. The main content area is titled 'Create OAuth client ID' and contains the following elements:

- A descriptive paragraph: "A client ID is used to identify a single app to Google's OAuth servers. If your app runs on multiple platforms, each will need its own client ID. See [Setting up OAuth 2.0](#) for more information."
- An 'Application type' dropdown menu set to 'Web application'.
- A 'Name' text input field containing 'Web client 2'.
- A note: "The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users."
- An information box: "The domains of the URIs you add below will be automatically added to your [OAuth consent screen](#) as [authorized domains](#)."
- An 'Authorized JavaScript origins' section with a help icon and a description: "For use with requests from a browser". It includes a '+ ADD URI' button.
- An 'Authorized redirect URIs' section with a help icon and a description: "For use with requests from a web server". It includes a '+ ADD URI' button.
- At the bottom, there are 'CREATE' and 'CANCEL' buttons.

- Go to the Blitz Identity Provider Admin console and add a provider that is of Google type.

<sup>31</sup> <https://console.developers.google.com>



---

3. Fill in the Identity Provider settings:

- Provider identifier;
- Provider name;
- The client identifier (`Client ID`) obtained from Google API Manager;
- The client secret (`Client secret`) obtained from Google API Manager;
- URL for authorization;
- URL for getting and updating the access token;

**Note:** If user access tokens are to be saved to the database, check `Remember tokens`. As a result, the tokens will be saved in the following cases:

- when a user logs in;
- when binding an external provider to User profile;
- when binding an external provider via REST API v2;
- when registering a user via an external provider;

- URL for getting user data;
- Requested scopes (`scope`) provided in [Google](#)<sup>32</sup>.

4. Customize binding rules.

5. In the “*Authentication*” section of the Management Console, enable the use of the Google Identity Provider authentication method.

---

<sup>32</sup> <https://developers.google.com/+/web/api/rest/oauth#authorization-scopes>

**Google basic properties**

Identity provider identifier   
Identity provider unique identifier. Is used only within Blitz Identity Provider

Identity provider name   
Identity provider display name. Is used only within Blitz Identity Provider

**Google Identity provider properties**

**Security**

Use section "Credentials" of the [Google API manager](#) to fill in the fields below. Do not forget to save the redirect URIs in the "Credentials" section.

Redirect URIs   
These links should be entered in the identity provider settings in order to process the user authentication results. Use the https scheme if you are using a secure connection.

URL for authentication

URL to get and update a token

Remember tokens

URL to obtain data

Client ID

Client secret [Change value](#)

---

**Scopes**

Requested scopes   
To add a scope enter its name and press Enter

Specify a list of scopes that should be requested from the identity provider [List of available Google scopes](#)

## Facebook Page 117, 1

To configure login via Facebook you need to follow the steps below:

1. Go to the [Facebook for developers console](#)<sup>33</sup>, in which you make the following settings:
  - add a new application by specifying its name, e-mail address for communication, and application category;
  - create an application identifier;
  - go to the application settings, section "General". In this section, specify the "Application domains" parameter (the parameter must correspond to the domain where Blitz Identity Provider) is installed) and add a site with a similar URL.
  - go to "Verify Application" and activate the item "Make Application «...» available to everyone".

<sup>1</sup> Meta is recognized as an extremist organization and is banned in Russia, while the activities of its social networks Facebook and Instagram are also banned in Russia.

<sup>33</sup> <https://developers.facebook.com/apps/>

2. Go to the Blitz Identity Provider Admin console and add a provider that is of Facebook type.

3. Fill in the Identity Provider settings:

- Provider identifier;
- Provider name;
- The application identifier (App ID) obtained from the Facebook for developers console;
- Application Secret (App Secret) obtained from Facebook's developer console;
- URL for authorization;
- URL for getting and updating the access token;

**Note:** If user access tokens are to be saved to the database, check Remember tokens. As a result, the tokens will be saved in the following cases:

- when a user logs in;
- when binding an external provider to User profile;
- when binding an external provider via REST API v2;
- when registering a user via an external provider;

- URL for getting user data;
- Requested scopes (scope) provided in Facebook<sup>34</sup>;
- Requested attributes provided by Facebook; it is acceptable to specify only those attributes provided by the selected permissions.

4. Customize binding rules.

<sup>34</sup> <https://developers.facebook.com/docs/facebook-login/permissions/>

- In the “Authentication” section of the Management Console, enable the use of the Facebook Identity Provider authentication method.

**Basic properties Facebook**

Identity provider identifier   
Identity provider unique identifier. Is used only within Blitz Identity Provider

Identity provider name   
Identity provider display name. Is used only within Blitz Identity Provider

---

**Facebook identity provider properties**

**Security**

To fill in the fields use the [Facebook for Developers](#) portal. Do not forget to save the specified domain in the Facebook application properties.

Application domain

OAuth 2.0 redirect URIs   
These links should be entered in the identity provider settings in order to process the user authentication results. Use the https scheme if you are using a secure connection.

URL for authentication

URL to get and update a token

Remember tokens

URL to obtain data

Application ID (App ID)

Application secret (App Secret) [Change value](#)

---

**Scopes and attributes**

Requested scopes   
To add a scope fill in its name and press Enter  
Specify the list of scopes that should be requested by contacting the identity provider. [List of available Facebook scopes](#)

Requested attributes   
To add a attribute fill in its name and press Enter. Specify a list of attributes that must be obtained from identity provider. The list of available attributes depends on what scopes are requested.

### Login via another Blitz Identity Provider setup

To configure login through an account of another Blitz Identity Provider (for example, one installed in another organization, hereafter referred to as a trusted Blitz Identity Provider) or other identity provider that supports OIDC, follow these steps:

- Open the admin console of the trusted Blitz Identity Provider (or have the administrator of another Blitz Identity Provider to do so) and perform the following operations:
  - go to “Appendices”;
  - click on the “Add an application” button;

- specify the application ID, name, and domain of the application;
  - save the application and proceed to customizing it;
  - select the OAuth 2.0 connection protocol;
  - specify a secret (`client_secret`), or leave the pre-populated option;
  - specify the prefix of the return link, which is the URL of the main Blitz Identity Provider to be logged in to;
  - configure the necessary scopes in the “*OAuth 2.0*” section.
2. Go to the Blitz Identity Provider admin console and add a provider that is of Blitz Identity Provider type.
  3. Fill in the Identity Provider settings:
    - Vendor Identifier;
    - Vendor Name;
    - The External Provider URI is the domain on which the trusted Blitz Identity Provider is installed;
    - The identifier (`client_id`) specified in the trusted Blitz Identity Provider settings;
    - The secret (`client_secret`) specified in the trusted Blitz Identity Provider settings;
    - Requested scopes, these scopes must be defined in the OAuth 2.0 section of the trusted Blitz Identity Provider;
    - Identifier - an attribute of the trusted Blitz Identity Provider that will be used as the user ID (ensures account uniqueness even if the attribute responsible for the username is changed);
  4. Customize binding rules.
  5. In the “*Authentication*” section of the Management Console, enable the use of the authentication method using Blitz Identity Provider identity provider.

Blitz Identity Provider basic properties
Blitz Identity Provider properties

### Security

To fill in these parameters contact the administrator of the external Blitz Identity Provider. Necessary information can be found in the properties of the connected applications (using OAuth 2.0 protocol). Also pass the administrator the following redirection URIs.

**Predefined redirect URLs (redirect\_uri)** `https://bip-dev1.reaxoft.ru/blitz/login/externaldps/callback/blitz/blitz_799/false https://bip-dev1.reaxoft.ru/blitz/profile/social/externaldps/callbackPopup/blitz/blitz_799`

These links should be entered in the identity provider settings in order to process the user authentication results. Use the https scheme if you are using a secure connection.

URL for authorization

URL for getting and refreshing tokens

Remember tokens

URL for getting data

Identifier (client\_id)

Secret (client\_secret)

---

### Scopes

Requested scopes

To add a scope fill in its name and press Enter

Specify the list of scopes that should be requested by contacting the identity provider. Contact the administrator of the external Blitz Identity Provider to get a list of available scopes

---

### Identification of accounts

Specify the unique attribute of external Identity provider attribute that will be used to indicate the account in the Blitz Identity Provider.

Identifier

## Account linking settings

Each identity provider's settings include a section called Account linking. You can use the settings in this section to define:

- rules for linking an external account to an account in Blitz Identity Provider;
- rules for matching attributes of an external account and an account in Blitz Identity Provider.

Two setting modes are provided: basic and advanced.

Linking an external account to an account in Blitz Identity Provider occurs in the following scenarios:

- The first time you log in using an external account, if it is not already linked to any account in Blitz Identity Provider.
- When binding in the User profile.

## Basic configuration

The basic configuration is performed using the Rule Builder. This mode is suitable for typical account linking and attribute mapping scenarios.

The following settings are provided:

- Allow one identity provider account to be bound to many accounts:
  - option selected - Blitz Identity Provider will allow an external account to be linked to multiple accounts in Blitz Identity Provider. When a user logs in with such an external account, they will be shown a selection of multiple linked accounts during the login process.
  - option not selected - Blitz Identity Provider will not allow an external account to be linked to Blitz Identity Provider account if that external account is already linked to another Blitz Identity Provider account.
- Prompt the user to enter login and password for binding if the account has not been identified:
  - option selected - the user will be prompted to identify and authenticate using an alternative method to bind an external account if the configured rules fail to find an account in Blitz Identity Provider.
  - option not selected - Blitz Identity Provider will not allow logins for users for whom no accounts could be mapped. If a logon process for external accounts is configured, the logon process will automatically start.
- Enable user registration:
  - option selected - the password entry form features a link that can be used to register in an external provider.
  - option not selected - proceeding to external provider registration in the password entry form is not possible.
- Only one account must be found for linking according to the specified matching rules:
  - option selected - if more than one account is found according to the matching rules, an error message will be displayed to the user.
  - option not selected - if more than one account is found according to the matching rules, there will be an option to continue the linking process.
- Require password entry if the account has been identified:
  - option selected - the user will need to authenticate to link their account to an external vendor account.
  - option not selected - the account will be automatically linked to an external vendor account.
- Customizing account identity rules - You can create rules to match identity attributes from an external account to identity attributes in Blitz Identity Provider. To create identity rules, you must use `${attr_name}` substitution strings, where `attr_name` is the name of the attribute received from the external identity provider. You can specify multiple attributes in a single rule. For example, the rule `email=${default_email-}` means that the `email` attribute in Blitz Identity Provider will map to the `default_email` attribute of the external account, provided that the `default_email` attribute is not empty. Multiple conditions can be specified (using the `+` add condition link to be met simultaneously and alternate rules can be added using the `+` add an alternative rule link).

**Account linking**

Basic configuration    Advanced configuration

**Identification of accounts**

Specify the rules for matching the Blitz Identity Provider and the social login provider accounts. The first time you log in through the social login provider, these rules will be used to search for an account in Blitz Identity Provider stores for binding with the social login provider account.

To create a rule, use substitution strings `${attr_name}`, where `attr_name` is the attribute's name from the social login provider account. You can specify several attributes in one rule. For example, the rule `CN=${name} ${surname}` means that the CN attribute will be formed from two attributes name and surname.

Allow one external user account to be linked to multiple Blitz Identity Provider accounts

Ask user to authenticate if no users were found

Allow user registration

Only one account by the configured matching rules should be found for binding

Require user authentication if account has been found

email = \${default\_email-} ✕

+ add condition

+ add an alternative rule

- **Block Attributes with rules for saving attributes.** For example, the `email=${default_email}` rule means that an attribute named `email` in Blitz Identity Provider will be populated with the value from the `default_email` attribute of the external account (for users who have used that identity provider). If the attribute has a `Master` checkbox checked, the attribute will be populated or updated each time the user logs in through the external Identity Provider. If the `Master` checkbox is unchecked, it will be populated only on the first logon that results in a credential bind.

**Attributes**

Specify how attributes used in the Blitz Identity Provider should be formed based on data from the identity provider. For each attribute should be set up a separate rule.

To create a rule use the denotation `${attr_name}`, where `attr_name` - is the attribute received from identity provider. You can specify several attributes in one rule. E.g. rule `CN=${name} ${surname}` defines that attribute CN should be formed by combining two attributes - `name` and `surname` - separated by a space. You can use rules to specify a constant or computed value. For example, the rule `uid=BIP-${random(4)}` will assign to the uid attribute the value of `BIP-XXXXXX`, where `XXXXXX` is a random value (a set of numbers and letters).

Attribute	Rule	Master
email	= \${default_email-}	<input type="checkbox"/>

+ Add attribute

- The User selection block defines the rules for displaying Blitz Identity Provider account found by the configured matching rules to the user. The `Username` setting defines the information displayed on the top line of the user card (the line intended to display the account name). For example, `${family_name-} ${given_name-}` specifies that the user's last name and first name (if filled in) will be shown on the top line. The `User identifier` setting determines the information displayed on the bottom line of the user card (the line intended to display the account ID). You can use value masking when customizing. For example, the `${phone_number&maskInMiddle(3,3)}` rule will display the middle numbers of a phone number as `*`.



## User Selection

User selection occurs if more than one account matches the match criteria or this external user is linked to more than one account

Username

Regular expression to display username

User identifier

Regular expression to display user identifier

- The Linked account block defines the rule of how a user's linked account is displayed in the user's external provider info in the admin console and user profile. The expression is formed based on the data received when a user logs in through an external provider.

## Linked account

Information about the linked account will be displayed to the user in the card of the external provider

Account name

The regular expression is formed from the data received when the user logs in through an external provider.

## Advanced configuration

In the case of the advanced configuration, the rules for account binding and attribute mapping are defined using a binding procedure in Java. This mode provides maximum configuration flexibility and is suitable for highly specialized account binding and attribute mapping scenarios.

## Account linking

Basic configuration

Advanced configuration

## Account linking procedure

For the binding procedure to work successfully, you must write a class in `Java` that inherits the abstract class `MatchingBlock`. The class name must be `Google_IGoogle`. The class must have a public `default` constructor. For security purposes, the class is loaded by a separate `class loader` with a limited list of `imports`. All necessary information is passed to function parameters.

```

1 package com.identityblitz.idp.federation.matching.dynamic;
2
3 import java.lang.*;
4 import java.util.*;
5 import java.text.*;
6 import java.time.*;
7 import java.math.*;
8 import java.security.*;
9 import javax.crypto.*;
10 import org.slf4j.LoggerFactory;
11 import org.slf4j.Logger;
12 import com.identityblitz.idp.federation.*;
13 import com.identityblitz.idp.federation.matching.*;
14 import com.identityblitz.idp.flow.common.api.*;
15 import com.identityblitz.idp.flow.common.model.*;
16 import com.identityblitz.idp.federation.matching.dynamic.*;
17 import java.util.function.Consumer;
18 import java.util.stream.Stream;
19 import java.util.stream.Collectors;
20 import org.codehaus.jackson.map.ObjectMapper;
21 import org.codehaus.jackson.type.TypeReference;
22 import com.identityblitz.idp.extensions.types.JsonObject;
23
24 import com.identityblitz.idp.federation.matching.*;
25 import com.identityblitz.idp.flow.common.api.HttpFactory;
26
27 /**
28  * The class is inherited from MatchingBlock and must have a default constructor to be instantiated correctly.
29  * The current generated implementation provides a strategy where users are not matched or updated.
30  */
31 public class Google_IGoogle extends MatchingBlock {
32
33     private final Logger logger = LoggerFactory.getLogger("com.identityblitz.idp.federation.matching.dynamic");
34
35     /**
36      * Iterative function determining the correspondence of internal accounts and identity provider accounts.
37      * At each iteration, the function can perform a find operation (found users will be passed in the next iteration)
38      * or terminate the operation with the following solutions:
39      * matched - matching users found;
40      * matchError - user matching error;
41      * matchByLogin - connect to a user who has successfully authenticated;
42      * refine - receives a list of users, asks for the password and connects with the user who entered the correct password;
43      * @param ctx - procedure context with the following fields:
44      *   iteration - procedure iteration number;
45      *   extAttrs - user attributes received from the identity provider;
46      *   sid - unique identifier of the external account.
47      * @param users - users.
48      * @return - one of the listed solutions
49      */
50     @Override public MatchResult match(MatchingContext ctx, List<MatchingUser> users){
51         return matchError(ctx, new MatchingError("not_matched", "User not matched"));
52     };
53
54     /**
55      * Returns attributes which can be updated or deleted.
56      * @param extAttrs - user attributes received from the identity provider.
57      * @param user - internal user.
58      * @param justMatched - an indication that the linking of internal accounts with external vendor accounts is established for the first time.
59      * @return - tuple with changeable and deleteable attributes. For example: change(JsonObj.empty(), Collections.<String>emptySet())
60      */
61     @Override public Tuple2<JsonObject, Set<String>> update(JsonObj extAttrs, MatchingUser user, Boolean justMatched, HttpFactory httpFactory){
62         return change(JsonObj.empty(), Collections.<String>emptySet());
63     };
64 }
65

```

Cancel

Save

## See also:

[Procedures for binding external user accounts](#) (page 217)

## 2.2.4 Customizing user services


Blitz Identity Provider provides web applications, with its help users can perform a number of transactions on their own:

1. Web application **User profile**. Allows you to perform a number of operations with your account, e.g. view/change your data, customize authentication methods, view recent events, change your password. If enabled, it is available at `https://{hostname}/blitz/profile`.
2. Web application **User registration**. If enabled, you can switch from the login page to the self-registration form (link “Don’t have an account? Register”).
3. Web application **Access recovery**. Allows a user to change his/her account password after passing the checks. If enabled, users will be able to navigate from the login page (link `:bdg-primary:“Forgot your password?”`) to the Restore Access form.

Configure these services in the Self-service section of the admin console.

**Attention:** The administrator of the admin console must personally check if JS-scripts placed on the login page are correct and make sure that content of the registration page and user profile is free of vulnerabilities.

### General settings

You can enable or disable the corresponding applications (services) on the main page of `:bdg-primary: “Self service”` section using the switch (  ). Please note that the switch only affects the display of links (e.g. Forgot your password?), while the availability of the service itself depends on whether the corresponding application has been installed by the administrator:

- `blitz-idp` – web application **User profile**,
- `blitz-registration` – web application **User Registration**,
- `blitz-recovery` – web application **Access recovery**.

The main page also allows you to configure the parameters that apply to all the self-services:

- confirmation code parameters sent to SMS - you can change the length of the code, its expiry time, and the number of attempts;
- confirmation code parameters sent by email - you can change the length of the code and its expiry time.

### Self-services

<b>Registration</b> <input checked="" type="checkbox"/>	<b>Recovery</b> <input checked="" type="checkbox"/>
Self-registration of users. <a href="#">Go to settings</a>	Self-service to recover access by sending a link to an email address. <a href="#">Go to settings</a>
<b>User profile</b> <input checked="" type="checkbox"/>	
The users tool to edit his data, including strong authentication, changing security settings. <a href="#">Go to settings</a>	

### General settings

Set the parameters of the confirmation codes sent via SMS and e-mail. These codes are used when registering users, to restore access to the account, as well as when changing the mobile number / e-mail address through the User Profile.

---

#### Confirmation code parameters sent to SMS

Length	<input type="text" value="6"/>
	Number of characters in the code
Expiry time	<input type="text" value="300"/>
	Number of seconds after which the code ceases to function
Number of attempts	<input type="text" value="3"/>
	Number of failed attempts to enter the confirmation code. If the number is exceeded, a new confirmation code is required

---

#### Confirmation code parameters sent by e-mail

Length	<input type="text" value="6"/>
	Number of characters in the code
Expiry time	<input type="text" value="2592000"/>
	Number of seconds after which the code is no longer valid.

[Save](#)

The subsections configure each self-service individually.

## User registration

**User registration** is a web application that allows a user to create his/her own account. Registration setup includes configuring the registration form, changing the service parameters and creating a registration procedure (optional).

### Registration form

The list of requested user data is defined by the HTML template. The template is a text file that is compiled using the [Twirl<sup>35</sup>](#) templating engine. It is necessary to set functions in the template that allow the user to enter data about himself when registering.

Examples of functions available in the template:

- `@attrInput("email", msg("reg.email"), Map("placeholder" -> "mail@example.com", "error-messages" -> msg("reg.email.wrong")), "input-type" -> "mail")` - displays on the page a field for entering the email attribute described in the system. `msg("reg.email")` is the name of the attribute, which is taken from the message file according to the current locale. If the input field is empty, it displays "mail@example.com" as a hint, and if the input is incorrect, it displays `msg("reg.email.wrong")` from the message file. The `input-type` equal to `mail` is set for the element;
- `@attrInput("family_name", "Last name", Map("placeholder" -> "Last name", "error-messages" -> "Error"))` - displays on the page the field for entering the user's last name into the `family_name` variable. This variable can be further used when executing the registration procedure.
- `@securityQuestionInput` – displays the input fields of the security question and the answer to the security question on the page;
- `@passwordsInput` - displays the password and password confirmation fields on the page;
- `@agreement` - displays the link to the User agreement;
- `@attrExpr` - the function, which allows to create a computed attribute (or assign a constant value to the attribute);
- `@submitButton` - displays the Register button.

An example of the template for registration:

```
@attrInput("family_name", "Last name", Map("placeholder" -> "Last name", "error-
↪messages" -> "Error"))
@attrInput("given_name", "Name", Map("placeholder" -> "Name", "error-messages" ->
↪"Error"))
@attrInput("phone_number", "Mobile phone number", Map("placeholder" ->
↪"+7(999)9999999", "error-messages" -> "reg.page.mobile.req.err.msg"))
@attrInput("email", "Email address", Map("placeholder" -> "name@example.com",
↪"error-messages" -> "reg.page.email.req.err.msg", "input-type" -> "mail"))
@passwordsInput
@agreement
@attrExpr("sub", "BIP-${&random(4)}")
@submitButton
```

**Tip:** To auto-generate the GUID of the created accounts, use the following formula `@attrExpr`:

```
@attrExpr("sub", "${&rUUID()}")
```

<sup>35</sup> <https://github.com/playframework/twirl>

The result of using the specified template in the interface of the web-application **User registration** is presented in figure:

To add a drop-down list to the registration form to select attribute values from the directory:

1. Create the `/etc/blitz-config/custom_messages/dics` directory on the `/project/` server;
2. Create a `/etc/blitz-config/custom_messages/dics/dic_name` file with the directory contents (instead of `dic_name`, specify the directory name, for example, `company_id`). Example of `company_id` file for the company selection drop-down directory:

```
001=Test company 1
002=Test company 2
003=Test company 3
```

The number in the directory will be written to the attribute value. The row in the directory will be shown to the user on the registration form.

3. Check the owner of the `dics` directory and the directory files in it. The owner must be `blitz:blitz`.

```
chown -R blitz:blitz /etc/blitz-config/custom_messages/dics
```

- In the `blitz.conf` configuration file, add the `dics` block to the `blitz.prod.local.idp.messages` block. In the `names` setting, list all directory names (a separate file with directory values must be created for each directory). For example:

```
"dics" : {
  "dir" : "custom_messages/dics",
  "names" : [
    "company_id"
  ]
}
```

- Restart the `blitz-registration` application.
- In the admin console, in the registration page template, add a line with the attribute filling from the directory:

```
@attrInput("company", msg("Company"), Map("dic" -> "company_id", "dic-default" ->
↵ "0", "sort" -> "key"))
```

### Registration service settings

The settings you can specify:

- store to save the account - select one of the configured storages (section Data sources) for saving the account;
- required for registration user attributes - attributes, the presence of which is necessary to complete the registration procedure. Mandatory user attributes do not need to be included in this list. It is possible to add several alternative rules. If the checkbox `Использовать условия из процедуры регистрации` is checked, the configured conditions are ignored and the conditions defined by the `isEnough` function from the registration procedure are applied.
- URL of the external enrollment service. If you specify this URL as a parameter, the user will be directed to this URL when he or she proceeds to the registration process (instead of the Blitz Identity Provider registration application).

The screenshot of a fragment of the registration settings page is shown in the figure below:

## Registration procedure

Registration procedure - Java code that implements the necessary checks after the user fills in the registration form. The following actions are possible during execution of the procedure:

- additional verification of the input data;
- conversion of the input data;
- saving attribute values in the storage;
- invoking the external REST services.

If required, convert the data entered by the user and then save them as attributes, in the registration page template you should use function `@attrInput` instead of `@textInput`.

## Changing the text in the User agreement

A link to the User agreement is located on the user registration page. The User agreement is located in the `assets.zip` archive located in the `assets` directory of the Blitz Identity Provider) installation in the archived `documents\user_agreement` directory.

To change the User agreement, unpack the `assets.zip` archive, replace `user_agreement_en.pdf` (Russian version) and `user_agreement_en.pdf` (English version) with required files and archive it keeping the original structure.

It is also possible to change the reference to the User agreement. To do this, [edit the](#) (page 234) line `reg.page.reg.action.agreement` and `setPswd.page.agreement`. This method is recommended if the User agreement is placed on an external resource, for example, as a separate web page.

## User profile

**User profile** is a web application in which the user can perform the following actions:

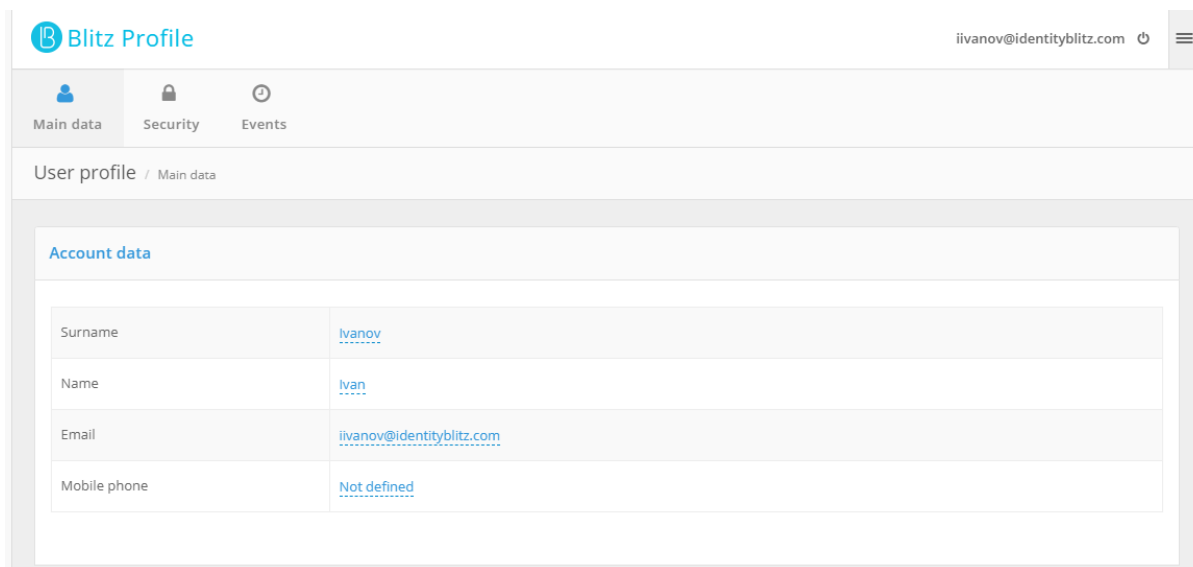
- view or edit their account data;
- view recent security events (e.g. login events);
- change password;
- view and configure the methods of login confirmation (two-factor authentication);
- view and configure the security keys;
- view bound social networking accounts; bind new external accounts; unbind unnecessary accounts;
- view the bound access devices, and unbind unnecessary devices;
- view and revoke data access permissions issued by applications;
- view security events.

Configuration of User profile includes configuring the way user attributes are displayed and change additional parameters.



## Displaying user attributes

The main page of myAlpari displays a block with account data. An example of this block is shown in the figure below.



The display of user data is defined by an HTML template. The template is a text file that is compiled using the [Twirl](#)<sup>36</sup> templating engine. In the template it is necessary to place functions that allow the user to enter and edit data about himself/herself in the User profile.

The following functions are available in the template:

- `@show(attrName)` - displays the attribute value;
- `@showStrings(attrName, values)` - displays the array value;
- `@editAsText(attrName, readableName, errorMsg)` - displays the value of the attribute and allows you to edit it (the `errorMsg` parameter is optional);
- `@editAsBoolean(attrName, readableName)` - displays the value of the logical type (true/false) of the attribute and allows you to edit it;
- `@editAsStrings(attrName, readableName, values)` - displays the value (array) of the attribute and allows you to edit it.

These functions use the following parameters:

- `attrName` – is the name of the attribute defined in the Data sources section;
- `readableName` - the name of the attribute, displayed to the user in the message (can be specified as attribute's identifier from a message file or as a text);
- `values` - values, in format `key - description`, where `key` is array value, the `description` - the readable value of the key (for example, `ListMap("a" -> "value a", "c" -> "value c")`), can be set as an identifier from the message file or as a text;
- `errorMsg` - error description, which is displayed in case of erroneous value input (can be set as an identifier from a message file or as text). About message files see. [Web interface texts](#) (page 234). It is recommended to use message files if you need to support multilingualism.

Example of functions:

<sup>36</sup> <https://github.com/playframework/twirl>

Listing 6: Email attribute display

```
@editAsText("email", "Email")
```

Listing 7: Displaying the phone\_number attribute with the ability to edit it

```
@editAsText("phone_number", "Mobile phone", "Error")
```

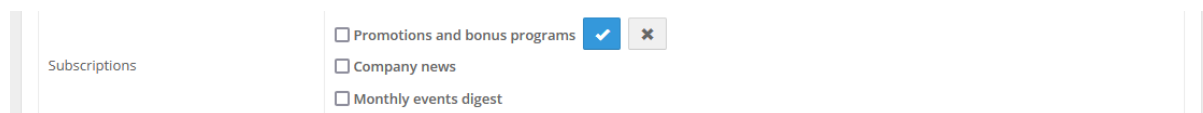
Listing 8: Displaying the boolean info attribute with the ability to edit it

```
@editAsBoolean("info", "Subscription")
```

Listing 9: Display an array of strings massiv with the ability to edit it (selection of values)

```
@editAsStrings("massiv", "Subscriptions", ListMap("a" -> "Promotions and bonus_
->programs", "b" -> "Company news", "c" -> "Monthly event digest"))
```

An example of displaying an array of strings in the interface of the **User profile** web application is shown in the figure:



### Additional parameters

The following parameters can be set as additional:

- welcome template - information that is displayed in the upper right corner of myAlpari. It is allowed to use substitution strings. For example, `${family_name} ${given_name}` will allow to display the surname and first name of the user;
- URL to follow after a successful logout from User profile;
- period of audit events displayed to users (in calendar months from the current date);
- template for displaying geodata in events (see [Geodatabase](#) (page 261)). The template can be composed of the following elements containing country, region, city and coordinate information: `${ip_ctr}`, `${ip_st}`, `${ip_ct}`, `${ip_lng}`, `${ip_lat}`, `${ip_rad}`
- functions available to users, i.e. functions that can be activated by the user from the User profile. It is possible to enable or disable the following functions:
  - password change;
  - setting up a security question;
  - security key management;
  - view and binding of social networks;
  - view of access devices;
  - view and revoke permissions;
  - view events;
  - HOTP generators binding;
  - TOTP generators binding;

- configuring login confirmation by SMS code;
- configuring push authentication;
- security key binding.

**Properties**

**Welcome template**   
It is displayed in the right upper corner of the User Profile

**Logout redirect uri**   
URL to which the user will be redirected after a successful logout

**Audit event depth**   
Number of full calendar months to view events

**Ip location template**   
It is displayed in the events

---

**Available to users functions**

- Password change
- Security question change
- Manage security keys
- View and link social networks
- View access devices
- View and revoke application permissions
- View events
- Binding of HOTP generators
- Binding of TOTP generators
- Configure the SMS confirmation
- Push-authentication properties
- Attach security keys

## Access recovery

### Console settings

The `Permissible attributes for search` setting of the access recovery service defines the attributes by which the account will be searched.

With the `Attributes for verification` setting, you can define which attribute values the user must additionally enter during the password recovery process to validate account ownership. Adding such verification complicates the password reset attack via multiple brute force in the Forgot Password Recovery form. On the main page, the user will be prompted for attributes to match (e.g. last name) and recovery will only be performed if the account found has an identical attribute value.

The `Verify that there are users who have permission to change password in the found account` option specifies that if the found user has a related (“parental”) account authorized to change the password for this user, a warning will be displayed when attempting to recover the password.

Possible recovery access contacts setting defines attributes with contacts (email addresses and/or mobile phone numbers) that will be used to restore access. Attributes with contacts should be defined in the Data sources section as an email address and a mobile phone number.

Using the settings *Total attempts* and *Blocking time when attempts are exceeded, in min.* you can limit the number of attempts to request sending and unsuccessful entry of confirmation codes sent by email and SMS for the account, if exceeded, the account will be temporarily restricted from password recovery.

*Need for additional verification* setting determines in which cases additional authentication should be performed during access recovery. Possible setting values:

- *Not required* – no additional authentication required;
- *According to user settings in Profile* – additional authentication is required if the user has enabled two-factor authentication for his account;
- *Always required* – additional authentication is always required;
- *Required if available* – additional authentication is required if at least one of the methods specified in the *List of methods* setting is available for a user.

If additional authentication is required, then in the setting *List of methods* you can select the available authentication methods to confirm the recovery of access: confirmation of the code received by e-mail, SMS, using the code generated by the TOTP application, using the answer to the security question.

The *Drop inactivity lock after restoring access* setting specifies that password recovery is allowed for accounts locked out due to long-term inactivity, and that the long-term inactivity lockout should be canceled after password replacement as a result of successful recovery.

### Recovery

**Search account**

**Permissible attributes for search**

Specify a list of user attributes that will be used for user search

**Attributes for verification**

Specify a list of user attributes, the values of which will be requested from the user for verification

Verify that there are users who have permission to change password in the found account

---

**Recovery methods**

**Possible recovery access contacts**

Specify a list of user attributes that correspond to possible user contacts

**Total attempts**

The total number of verification code submissions and verification code attempts that will cause the authentication method to be temporarily blocked

**Blocking time when attempts are exceeded, in min.**

During the specified time, the authentication method will be unavailable to the user

**Additional verification**

**Need for additional verification**

---

**After recovery operations**

Drop inactivity lock after restoring access

### Form texts

After defining the set of [verification attributes](#) (page 134), you must specify the corresponding texts in the access recovery form. To do this, use the [standard algorithm](#) (page 234). Add texts for the following lines:

- `recovery.page.verify.<attribute name>.label`: name of the field for entering the attribute value;
- `recovery.page.verify.<attribute_name>.placeholder`: text inside the field for entering the attribute value.

Listing 10: Example of setting texts for the `phone_number` and `family_name` attributes

```
recovery.page.verify.phone_number.label=Mobile phone number
recovery.page.verify.phone_number.placeholder=Enter your phone number
recovery.page.verify.family_name.placeholder=Last name
recovery.page.verify.family_name.placeholder=Enter your last name
```

## 2.2.5 User administration

This section is devoted to user administration in Blitz Identity Provider.

### User account management

In the section Users of the admin console Blitz Identity Provider administrator can perform the following operations:

- [search for user accounts](#) (page 138);
- [add a user account](#) (page 139);
- [view and edit user account attributes](#) (page 141);
- [reset user sessions](#) (page 141);
- [change the password of the user account](#) (page 142);
- [view and unlink accounts of external identity providers](#) (page 142);
- [link devices for two-factor authentication](#) (page 142);
- [view the groups in which the user is included, manage the user's membership in groups](#) (page 144);
- [view, link, delete user security keys](#) (page 148);
- [view user account rights, assign and revoke rights](#) (page 145);
- [view permissions granted by the user to applications](#) (page 148);
- [view and delete stored devices](#) (page 147);
- delete the user account.

The general view of the User Data Management page is shown in the figure below.

The screenshot displays the 'User search' interface. At the top, there is a search bar containing 'mail@example.com' and a 'Find' button. Below the search bar, there is a link 'Create a user account...'. The main content area is divided into two sections: 'User accounts' and 'User data'. The 'User accounts' section shows a list of users, with one user selected: 'built-in: ivanov@example.com' and 'Иван Иванов'. The 'User data' section shows a form with the following fields: name (Иванов), username (ivanov), surname (Иван), mail\* (mail@example.com), uid (ivanov@example.com), mobile (+7(910)1234567), and email (mail@example.com). Below the 'User data' form is a 'Change password' section with a 'New password' field and a 'Generate a password' checkbox. The 'Save' and 'Change' buttons are located at the bottom right of their respective sections.

## User search

To search for users, enter the user ID and click the “Search” button. The attribute is used as the displayed identifier, defined in the “Data sources” section as the base identifier, as well as attributes marked as search attributes.

The list of users found contains:

- identifier of the found user;
- store where user was found;
- user name, configured in the “Data sources” section.

Clicking on any of the found accounts opens the information details of the user.

Also available:

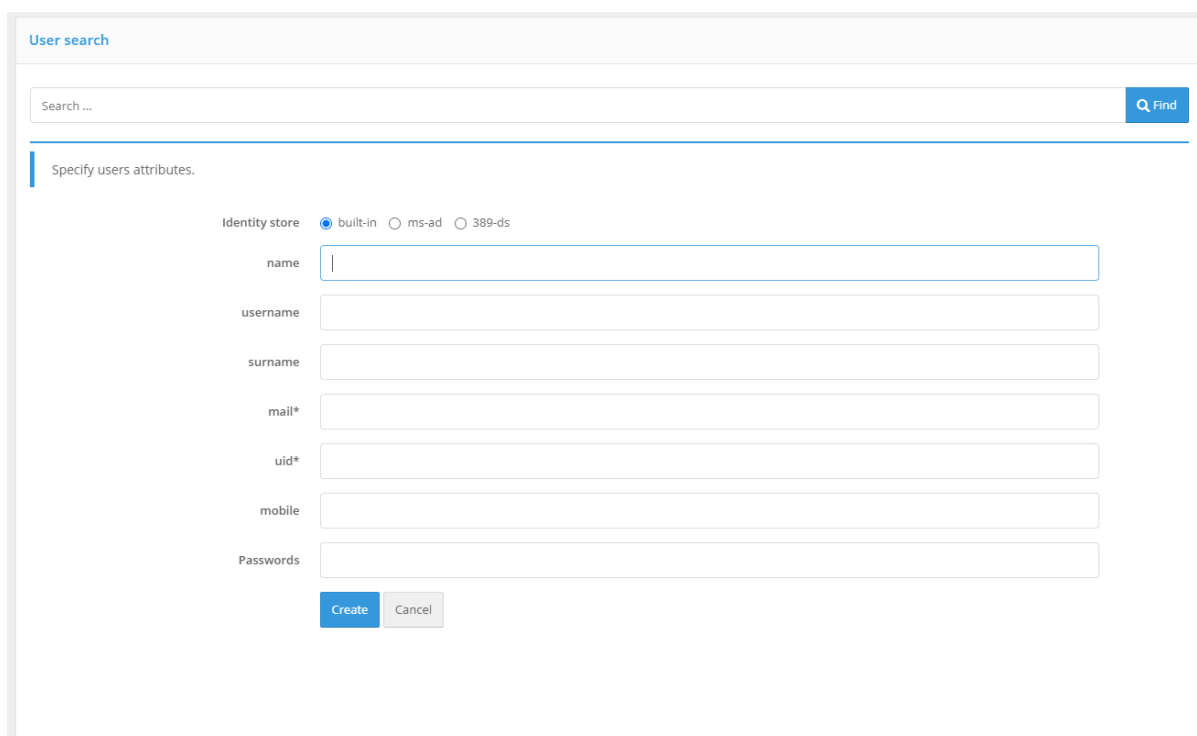
- when you click link copy button, the link to the found user will be copied to the clipboard;
- the link “Security events” allows you to quickly view security events for the current day, in which the found user appears as an access object.

## Adding a user

To add a new account, click on the “*Create a user account...*”. In the opened window:

- specify the store where user data should be saved;
- set all required attributes;
- click on the “*Create*” button.

**Important:** During account creation, you should consider the datastore configurations and restrictions. For example, if the record is saved to an LDAP directory, all mandatory attributes must be filled in, attribute uniqueness restrictions must not be violated, etc. From the Blitz Identity Provider point of view, only the identifier and mandatory attributes are mandatory (the corresponding attributes are marked with an asterisk (\*)).



The screenshot shows a "User search" dialog box. At the top, there is a search bar with the text "Search ..." and a "Find" button. Below the search bar, the text "Specify users attributes." is displayed. Underneath, there are three radio buttons for "Identity store": "built-in" (selected), "ms-ad", and "389-ds". Below these are several input fields: "name", "username", "surname", "mail\*", "uid\*", "mobile", and "Passwords". At the bottom of the dialog, there are two buttons: "Create" and "Cancel".

## View and edit user attributes

To display information on any found user, click on the identifier of the user. It contains the attribute values that were defined in the section “*Data sources*”, as well as linked accounts of external identity providers, user devices, security keys, etc.



**User search**

mail@example.com Find

[Create a user account...](#)

---

**User accounts**

built-in: ivanov@example.com  
Иван Иванов

**User data**

name	<input type="text" value="Иванов"/>
username	<input type="text" value="ivanov"/>
surname	<input type="text" value="Иван"/>
mail*	<input type="text" value="mail@example.com"/>
uid	<input type="text" value="ivanov@example.com"/>
mobile	<input type="text" value="+7(910)1234567"/>
email	<input type="text" value="mail@example.com"/>

[Save](#)

**Change password**

New password   Generate a password

[Change](#)

**Accounts of external systems**

Accounts of external systems not found

**The required authentication level**

You can set the authentication level required for this user. Option "default" means that the user must have a level specified in section "Authentication".

required level

[Save](#)

You can perform the following operations in the user card:

- edit user attributes;
- reset user sessions;
- change the password;
- view the list of bound accounts of external authentication providers, unbind external accounts;
- change the required authentication level for the user;
- bind or remove authentication devices: one-time password generators and mobile apps to receive push notifications;

- view the groups the user is included in;
- view the user's rights and the rights that are available for that user;
- view and delete saved user's devices and browsers;
- view, add, and delete user security keys;
- view and delete scopes granted to applications.

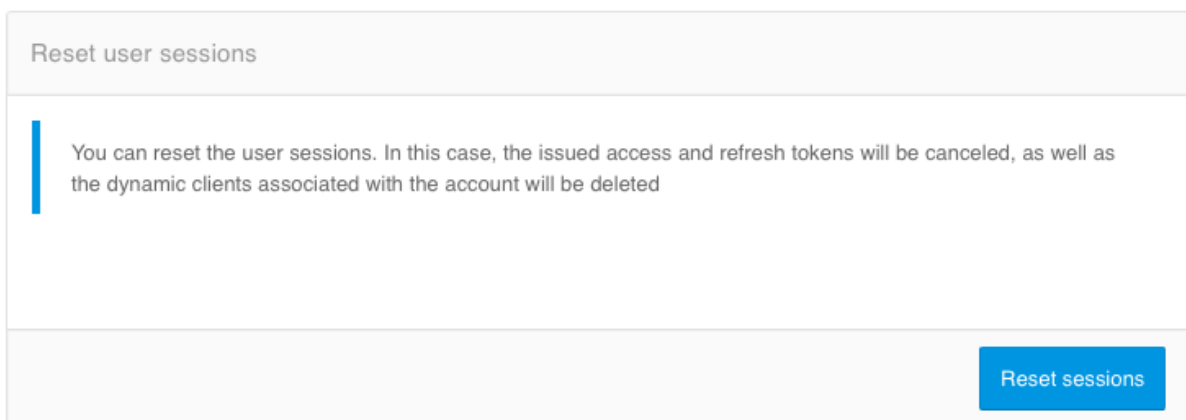
### Editing attributes

Administrators can change any attribute of the user when viewing the card of the selected user account. Note, when editing an account, be aware of the datastore configurations and restrictions to which the record is being written.

Note that changing data via the attribute editing interface disregards the rules used in the user self-registration process. For example, changing the e-mail address or cell phone number does not require confirmation.

### Resetting sessions

To reset user sessions, use the button Reset sessions in the block Resetting user sessions.



When resetting user sessions, the following actions are performed:

- security tokens issued to applications by the user (access tokens, update tokens, identification tokens) become invalid – when calling the introspection service with such tokens in Blitz Identity Provider, the service returns that the token is invalid;
- in devices stored for the user, the flags of trusted devices and the storage of long sessions on them are removed;
- dynamic `client_id/client_secret` pairs issued for mobile applications linked to the user account are canceled;
- the SSO sessions stored in the user's browser become invalid, so that at the next request from the identification applications in Blitz Identity Provider, a new identification and authentication will be requested.

## Changing the password

To change the password, use the block Changing the password. You can enter a new password manually, or generate it – to do this you, need to leave a checkbox Generate a password. The new password will be displayed in the information block of the successful operation. When changing the password, you can also set a checkbox Reset sessions, then the user’s sessions will be reset simultaneously with the password change.

When setting a new password manually, take into account the limitations of the password policy of the store where you are saving the password.

Change password

**New password**

Generate a password

Reset sessions

## View and unlink external providers

In the block “Linked accounts of external systems”, you can view the list of accounts of external identity providers (social networks, etc.) linked to the account of the found user. Each linking is characterized by a unique identifier, where the last part is the internal identifier of the account in the corresponding identity provider. If necessary, you can remove the link to an external account.

Accounts of external systems

**Google**

Account with identifier google:google\_1:102862191524241857221 has been bound

×

**Apple**

Account with identifier apple:apple\_1:002021.d8091805aa194cb19a475a92aad21e4c.1629 has been bound

×

## Binding devices for 2FA with a one-time password

The administrator can bind a two-factor authentication tool to the selected user account. For example, a hardware HOTP/TOTP generator can be bound by serial number, or a mobile application that generates TOTP codes can be bound to the account by QR code.

A device for generating one-time passwords (HOTP)

Serial number

Serial number of the device for generating one-time passwords

Value 1

Value 2

[Attach](#)

Time-based one-time password generator (TOTP)

Application name

Encryption algorithm

Password length


The number of symbols in a one-time password

Password refresh time

A new password will be generated when the time (in seconds) expires

Secret

Secret is Base32 encoded



[Save](#)


## Binding Duo Mobile

To make authentication via Duo Mobile, it is necessary to bind the mobile application to the user account. The recommended scenario is that the user binds their mobile app to the user's account in the "User profile" web application.

Another way to bind is via the admin console. To do this, it is necessary to find the necessary account in the "Users" section and the settings block "Duo Mobile application (QR code)". In this block, click on the "Attach Duo Mobile" button, then scan the displayed QR code with the Duo Mobile mobile application.

Duo Mobile application (QR code)

Scan the QR code with the Duo Mobile user app and click "Save".



duo://KOj14IDEE556dm8Rq6ac-  
YXBpLWFiMTZIMDhjLmR1b3NIY3VyaXR5LmNvbQ

Save





## Group Membership Management

If the user is included in groups, this information will be displayed in the block "Group membership". The following data will be displayed for each group:

- group identifier;
- group attribute values.

Group members	
Identifier	Group data
1147746651733	<p>OGRN: 1147746651733</p> <p>orgName: IT Company</p> <p>INN: 0000000000</p> <p>orgPhone: +7(495)1234567</p>

To exclude a user from a group using the delete button or add a user to another group use the “Add to group” link. To add a user to a group, you will need to enter the value of the attribute identifying the group, click the “Search” button, select the appropriate group from the list of found groups, and click the “Add” button.

Членство в группах		
Идентификатор	Данные группы	
admins	name: Администраторы	
power_users		
Идентификатор	Имя	
power_users/roles	Power Users	
		


### Viewing, assigning, and revoking rights

If the user has rights to the user from applications or other accounts, this will be displayed in the “Rights of subjects on user” block. If the user has rights over objects, such as other accounts, this will be displayed in the “User object permissions” block.

Each right is characterized by the following parameters:





- object identifier;
- name;
- right.

Права субъектов в отношении пользователя

Идентификатор	Имя	Право	
test-system	test-system	Назначать права	

[Назначить права](#)

Права пользователя в отношении объектов

Идентификатор	Имя	Право	
_blitz_profile	custom.app.name._blitz_profile	Менять пароль	
_blitz_profile	custom.app.name._blitz_profile	rights.right.SUPPORT	
isergeev@domain.com	Сергеев Иван Петрович	rights.right.TEST	
_blitz_console	custom.app.name._blitz_console	Назначать права	

[Назначить права](#)

You can revoke an access right using the delete button next to the access right. You can assign an access right using the *“Assign rights”* link. In this case you will have to select the assigned access right from the list, select the type of subject (user or application) or object (user, group or application), find and select the subject/object.

Rights of subjects on user

No rights found

Right

Subject type

Subject search

### User object permissions

No permissions found

**Right**

**Object type**

**Object search**

### Memorized devices and browsers

The administrator can view the devices and browsers the user has logged in using their account from. The description of devices includes:

- an indication of whether the device has a login session saved and whether the device is trusted. The indication is color coded:
  - gray - the login session is not saved on the device and the device is not trusted;
  - yellow - the login session is not saved on the device, but the device is trusted;
  - blue - a login session is saved on the device, but the device is not trusted;
  - green - the login session is saved on the device and the device is trusted.
- the name and operating system version of the device, determined from `UserAgent`;
- the browser name and version defined based on `UserAgent`;
- the date and time of the last login from this device and browser;
- The IP address of the user that was determined the last time the user logged in from this device and browser.

### User devices

No user devices found



## Security keys

The administrator can view the list of security keys (Passkey, WebAuthn, FIDO2, U2F) registered for the user account. For each security key, the following are listed:

- key name;
- date and time of key registration;
- scope of application (for Passkey and FIDO2 - for login and for login confirmation; for U2F - for login confirmation only);
- date and time of the last use of the key.

The screenshot shows a panel titled "Security keys". Inside the panel, there is a message box that says "No user security keys". Below the message box, there is a blue link labeled "Add key".

The administrator can register a new security key using the “Add key” link. In a typical usage scenario, security keys are added by the user himself at the moment of login (onboarding) or via the User profile.

The screenshot shows a form for adding a new security key. At the top, there is a label "Enter the key name". Below this, there is a text input field with the label "Key name" to its left. At the bottom of the form, there are two buttons: a blue "Create" button and a grey "Cancel" button.

The ability for an administrator to add a key can be useful in the following scenarios:

- The administrator personally issues users a hardware FIDO2/U2F key and binds it to the account. Two-factor authentication is used to access the company’s applications.
- The administrator needs the ability to log in to the user account for technical support purposes. Resetting the password from the account will inconvenience the user - instead, a security key can be registered and used to log in. All actions to register and delete security keys are logged as security events.

## Permissions granted to applications

The administrator can view a list of permissions granted by the user to applications.

Each permission is described by:

- identifier of the application;
- list of permissions (*scope*);
- date when the permissions were granted.

Application permissions

**new\_test\_app**  
scopes openid, profile  
date: 01.04.2021

✕

## Managing user groups

### Enabling the display of groups in blitz.conf

If Blitz Identity Provider is configured to work with user groups, Groups section appears in the admin console.

To enable the ability to view user groups, you must add `blitz.prod.local.idp.groups` following settings block:

```
"groups": {
  "profiles": [
    {
      "type": "mirror",
      "id": "orgs",
      "groupStore": "389ds",
      "attrsMap": {
        "name": "displayname",
      },
      "filter": "objectClass=group"
    }
  ],
  "stores": {
    "list": [
      {
        "type": "ldap_based",
        "id": "389ds",
        "desc": "Группы",
        "ldapStore": "389ds",
        "baseDN": "ou=external,ou=groups,dc=test",
        "searchScope": "SUB",
        "idAttrName": "cn",
        "membersAttrName": "uniqueMember",
        "memberOfAttrName": "memberOf",
        "newGroupAttrs": [
          {
            "attr": "objectclass",
            "format": "strings",
            "value": "top,groupOfUniqueNames,group"
          },
          {
            "attr": "dn",
            "format": "string",
            "value": "cn=${id},ou=external,ou=groups,dc=test"
          }
        ]
      }
    ]
  }
}
```

Specifics of settings:

- in `profiles.groupStore`, `stores.list.id`, `stores.ldapStore` must be the identifier of the LDAP directory used to store users;
- in `profiles.attrsMap` and `stores.list.idAttrName` must contain group attributes (class groups), e.g. `name`. Attribute names can be named differently if desired, only LDAP attributes of type `String` are supported;
- in `stores.list.baseDN` you should check (and correct if necessary) the path for storing organizations in LDAP. If the path is corrected, also adjust the `"value": "cn=${id},ou=external,ou=groups,dc=test"` setting accordingly.

## Working with groups

In the section Groups you can search for groups by one of the configured attributes, edit groups, create and delete groups, and manage user membership in groups.

For each group found, its attributes are displayed. In addition, Group Members block displays all users included in the group. For each user the following is displayed:

- user identifier;
- user name - according to the template defined in the Data sources section (`Username` on console).

**Group search**

Profile	Attribute	Value	
orgs	id	1147746651733	<a href="#">Find</a>

**User groups**

1147746651733

**Group data**

OGRN	1147746651733
orgName	IT Company
INN	0000000000
orgPhone	+7(495)1234567

**Group members**

id	Name
user	

You can edit group attributes, delete a group, add users to a group using the link [Add user...](#), remove a user from a group, and create new user groups using the link [Create a group...](#)

Adding a user to a group:

Члены группы

Идентификатор	Имя пользователя	
6647dc35-0c4f-4054-a7e7-fae41b011b4f	Петров Иван	
18539368-f59f-4ef1-8f34-3389272fa8bd	Васильев Дмитрий	

Иванов 🔍 Найти

Идентификатор	Имя пользователя
a9072d2b-9e89-45a5-9447-d0d126ba0332	Иванов Александр
854436f6-af58-4a3f-8cb7-c2c441eb4a76	Иванов Сергей

Добавить Отмена

Профиль  grps

Идентификатор группы

new\_grp

name

Создать

Отмена

## Access rights management

To maintain a directory of access rights in Blitz Identity Provider, use the “*Access rights*” section of the admin console. Access rights can be used to control user access to applications, to control the invocation of protected REST services by applications, and can be requested and used by applications to control user access to application functions.

**List of access rights**

Set the access rights that can be assigned to users, groups, or applications.

Name	Description	
Expert		
external	external_system	
Test	Test	
User		
Support		
Operator		
Administrator		

[+ Add access right](#)

[Save](#)

## 2.2.6 Notifications and sending messages

To configure notification settings and connect to messaging systems, use the *"Communication settings"* section of Blitz Identity Provider admin console. In this section, you can configure notifications and connections to:

- SMS delivery service;
- push notification service;
- SMTP-server.

To configure notifications on the main page of the section you need to:

- select a channel for recovery (e-mail, cell phone) and specify an attribute with the value of this contact. The attribute is specified using a regular expression, for example, `phone_number` means that the information will be sent to `phone_number`;
- select the events for which you want to send notifications. The following events can be notified:
  - input from an unknown device;
  - password change;
  - password change in dependent account;
  - password recovery;
  - password recovery in the dependent account;
  - bind the social network account;
  - unbind the social network account;
  - configuring two-factor authentication;
  - changing the login confirmation mode;
  - obtaining the right to change password in the dependent account;

- granting the right to change password;
- revoking the right to change password in the dependent account;
- revoking the right to change password;
- registration of user account;
- adding a new security key;
- deleting the security key.

**Notifications**

Configure notifications and users will be notified of various security events

**Notification methods**

Notify by	Attribute with contact	
SMS		✕

+ Add notification method

**Notify users of events**

- Sign in from an unknown device
- Change password
- Change password in dependent account
- Access recovery
- Password recovery in the dependent account
- Bind the social network account
- Configuring the two-factor authentication
- Obtaining the right to change the password in the dependent account
- Granting the right to change the password
- Revoking the right to change the password in the dependent account
- Revoking the right to change the password
- Registration of user account

Save

### Configuring connection to SMS gateway

Blitz Identity Provider requires the ability to send SMS if the following functions are used:

- authentication based on SMS confirmation code (first and second factor);
- notifications about important security events via SMS;
- changing the mobile phone number via "User Profile";
- password recovery using the mobile phone as an account proof of ownership;
- confirmation of the mobile phone number during user registration.

The settings are configured in Blitz Identity Provider admin console in "Communication settings" section.

**SMS delivery service properties**

Delivery protocol: HTTP-GET  
Message delivery protocol

Use substitution strings to form the service URL:  
 \${login} - login for the service access  
 \${password} - password to the service access  
 \${message} - SMS-message (required parameter)  
 \${mobile} - mobile phone number (required parameter)  
 URL example: https://test.sms-gateway.ru/send.php?login=\${login}&psw=\${password}&phones=\${mobile}&mes=\${message}

URL: https://smc.ru/sys/send.php?psw=\${password}&login=\${login}&phones=\${mobile}&mes=\${message}&charset=utf-8

Login: reaxoft  
Login to access the message delivery service

Password: [Change value](#)  
 Use Basic HTTP authentication

Headers:   
HTTP request headers. Each header is described on a separate line. The title and the value of the header must be separated by the symbol :

Template for successful operation: ^OK.\*  
A regular expression that defines a successful sending of the message, e.g. ^OK.+

Template for an error: ^ERROR.\*  
A regular expression that defines an error while sending the message, e.g. ^ERROR.+

The following settings must be configured:

- type of delivery protocol (GET or POST);
- SMS gateway URL - set in the form of a pattern to form a request to the SMS gateway to initiate sending of SMS by it. Example of setting for SMS gateway:

```
https://smc.ru/sys/send.php?psw=${password}&login=${login}&phones=${mobile}&mes=${message}&charset=utf-8
```

- login and password for access to the SMS gateway. Login and password can be passed as GET request parameters or as HTTP request header (HTTP Basic authentication scheme);
- HTTP request header to the SMS gateway;
- a template for checking the response from the gateway indicating successful sending. It is specified as a regular expression;
- a template for checking the response from the gateway indicating an error of sending a message. It is specified as a regular expression.

### Connection to the service of sending push notifications

Push notification settings are configured in the Admin console web application in the “Messages” section.

The following settings must be configured:

- type of delivery protocol (GET or POST);
- URL of the push notification service, for example:

```
http://api.system.ru/json/v1.0/communication/mobile/push
```

- data - a message passed in the body (body) of the request, for example:

```
{"token":"${password}","title":"${title}","body":"${message}","msisdn":$
↪{subscriberId}}
```

- login and password to access the service. Login and password can be passed as GET request parameters or as HTTP request header (HTTP Basic authentication scheme);
- HTTP request header;
- a template for checking the response from the service, indicating successful sending. It is specified as a regular expression, for example:

```
.\\"errorCode\":0.+
```

- a template for checking the response from the service that indicates an error in sending a message. It is specified as a regular expression, for example:

```
.\\"errorCode\":[1-9].+
```

An example of setting up integration with a push notification service is shown in the figure below.



### Configuring the push-notification service

Delivery protocol  Message delivery protocol

When generating the URL, HTTP request headers and body, use substitution strings:

- `${login}` - service access login
- `${password}` - service access password
- `${message}` - message text (required parameter)
- `${title}` - message title (required parameter)
- `${subscriberId}` - push user ID (required parameter)

URL

Data

Data transmitted in the body of the HTTP request

Login  Login to access the message delivery service

Password [Change value](#)

Use Basic HTTP authentication

Headers  HTTP request headers. Each header is described on a separate line. The title and the value of the header must be separated by the symbol `:`.

Template for successful operation  A regular expression that defines a successful sending of the message, e.g. `^OK+`

Template for an error  A regular expression that defines an error while sending the message, e.g. `^ERROR+`

### Configuring the connection to the SMTP gateway

In Blitz Identity Provider, you must configure the ability to send email messages if the following features are used:

- Notification of important security events by email;
- changing your electronic signature email address via your *“User Profile”*;
- recovering a forgotten password using email as a channel to confirm account ownership;
- confirmation of the e-mail address when registering a user account.

The settings are configured in Blitz Identity Provider admin console in *“Communication settings”* section.

SMTP-server properties

Host

Port

Use TLS

Sender   
Sender's email address

Login   The same as sender's addressd  
Login of the account to connect to the SMTP server

Password    
Password for connecting to the SMTP server

The following settings must be configured:

- SMTP gateway host name;
- SMTP gateway host port;
- whether or not it is necessary to use TLS for a secure connection to the gateway;
- sender's email address;
- account name at the SMTP gateway on behalf of which Blitz Identity Provider will send the email (if the account name is the same as the sender's email, then check the appropriate checkbox);
- password for the SMTP gateway account on behalf of which Blitz Identity Provider will send email;
- settings - additional configuration parameters of interaction with [SMTP gateway](#)<sup>37</sup>.

## 2.3 Access to applications and network services

### 2.3.1 Registering applications in Blitz Identity Provider

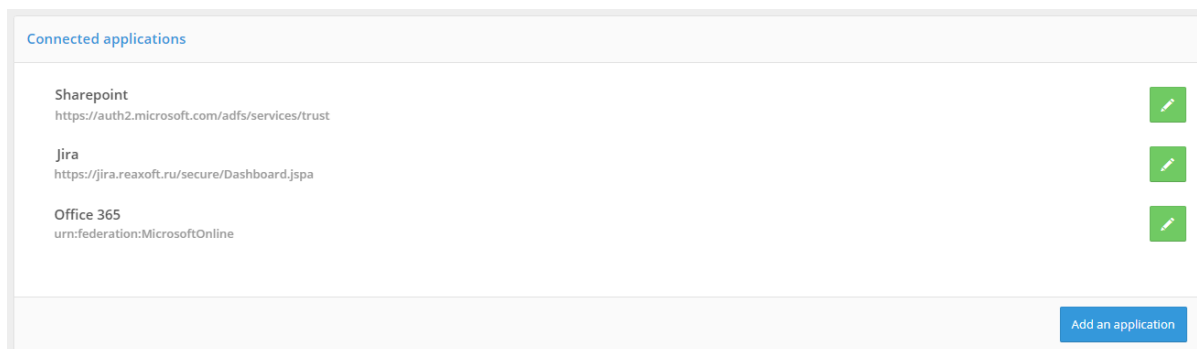
#### About applications

Application registration in Blitz Identity Provider is required so that applications can use the services provided by Blitz Identity Provider:

- request user identification and authentication;
- invoke the Blitz Identity Provider REST services.

Applications are managed in the section Applications of the Admin Console.

<sup>37</sup> <https://javaee.github.io/javamail/docs/api/com/sun/mail/smtp/package-summary.html>



### Creating a new application account

To connect a new web application, go to the Applications section of the console and select Add application. This action will launch the new application connection wizard, which includes the following steps:

#### Step 1. Basic settings

It is required to specify the identifier of the application to be connected (when connecting via SAML protocol, the identifier corresponds to `entityID`, when connecting via OAuth 2.0 - `client_id`), its name and domain, i.e. the URL where this application is available.

---

**Important:** When specifying the identifier for OAuth 2.0 it is not allowed to use colon and tilde.

---

The name of the application is then used by Blitz Identity Provider to display on the login page when the application initiates a request for user identification.

The application domain is used when a user needs to be redirected to the application from Blitz Identity Provider web pages. The redirection is done to the specified domain or to a specialized `redirect_uri` passed in the interaction with Blitz Identity Provider, but it is verified that `redirect_uri` corresponds to the domain specified in the application configuration.

## Step 2. Specify the application start page and select the login page template

In the “*Application home page*” field it is recommended to specify the application login link that initiates the identification and authentication request.

In the `Page Template` list, you must select which template should be used to display the login page when a user attempts to access this application. Instructions for creating a new template can be found [here](#) (page 221).

If necessary, you can specify the identifier encryption key (`privacy domain`). Creating a privacy domain ensures the uniqueness of the user identifier received by the application as a result of authentication, i.e. this identifier will be unique, but specific to this application. In other words, if a request for user data is initiated by an application from a different privacy domain, it will receive a different value of the user ID. Clicking on the field will display the previously configured encryption keys, with the option to set a new one. Applications that share a common encryption key will receive an identical User ID.

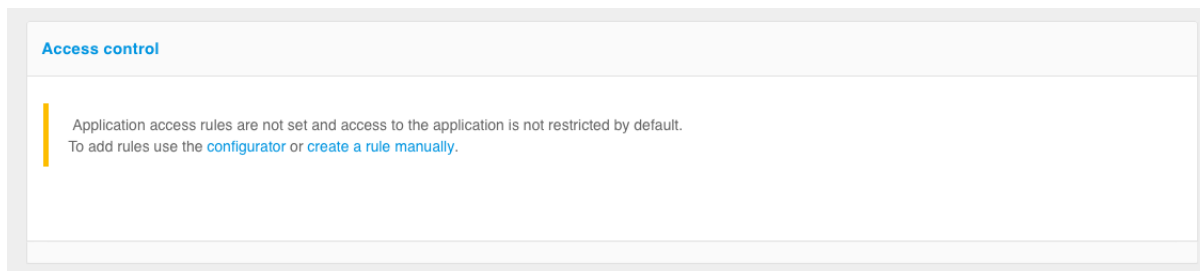
At this step, you can also set tags to further use them when customizing the application logic, e.g. to analyze them in a [login procedure](#) (page 214).

### Application settings

Identifier (entityID or client_id)	<input type="text" value="test-app"/>
	<small>Application identifier. Used for identifying application within the SAML (corresponds to entityID) and OAuth 2.0 (corresponds to client_id) protocols.</small>
Name	<input type="text" value="test-app"/>
	<small>Human-readable application name. Is used only inside Blitz Identity Provider.</small>
Domain	<input type="text" value="test-app"/>
	<small>Usually a link to the application's start page, e.g. http://testdomain.ru/. If TLS-authentication is used, then the domain should correspond to the domain specified in the certificate.</small>
Application start page	<input type="text"/>
	<small>Link to the application start page, e.g. http://testdomain.com/private. When logging in using SAML, it is used as a link to go to the application in case the login page is opened from the browser history</small>
Identifier encryption key	<input type="text"/>
	<small>If the key is specified, the user ID for the application will be encrypted using this key. The key value can be selected from a list. You can also assign a new key by typing it in the search box and pressing Enter</small>
Page template	<input type="text" value="default"/>
	<small>Page template determines the login page appearance. If the template is not specified, then the default template is used.</small>
Logout redirect uri prefixes	<input type="text" value="To add a new prefix enter it and press Enter"/>
	<small>A prefix is used to check the redirect uri. If the logout request includes an <code>post_logout_redirect_uri</code> that doesn't correspond to any prefix, then the logout is rejected</small>

### Step 3. Configure application access rules

You can configure the rules that Blitz Identity Provider uses to decide whether or not to allow a user into an application.



Access control rules can be added using the configurator or manually using RQL expressions (see figures below). In the rules it is possible to check that the user is included in the required user group (setting `Groups` in the configurator or rule `contains (grps, GRP1, GRP2, ...)`), has the required access right (setting `Authority` in the configurator or rule `contains (rights.its.SYSTEM, RIGHT_1, RIGHT2, ...)`) or has the specified attribute value (setting `Approval` in the configurator or expression with `attribute`).

**Access control**

In case the access is denied to the application, redirect the user to the access deny page. If not checked, then the user will be redirected to the application with an error according to the authentication protocol

---

**Configured application access rules**

This block specifies access rules for application using Resource Query Language (RQL). For successful access it is enough that at least one rule is fulfilled. Rules can be set manually or the configurator to create simple rules can be used. The rule also has a name and a description. If the name is specified, then it will be saved to the audit, otherwise the RQL textual representation will be written to the audit. The description is not used in processing and may contain any annotations associated with the rule.

[Data and expressions available in rules](#)

**Groups**   
User must be a member of the specified group

**Rights**  to application   
The user must have the specified right on the specified object. The object can be an applications, a group or a user

**Claim**  =   
User must have the specified claim value

**Rule name**   
If a rule has a name, it will be saved to audit

**Access control**

In case the access is denied to the application, redirect the user to the access deny page. If not checked, then the user will be redirected to the application with an error according to the authentication protocol

---

**Configured application access rules**

This block specifies access rules for application using Resource Query Language (RQL). For successful access it is enough that at least one rule is fulfilled. Rules can be set manually or the configurator to create simple rules can be used. The rule also has a name and a description. If the name is specified, then it will be saved to the audit, otherwise the RQL textual representation will be written to the audit. The description is not used in processing and may contain any annotations associated with the rule.

[Data and expressions available in rules](#)

Rule	Name	Description	Enabled	
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input checked="" type="checkbox"/>	<input type="button" value="✖"/>

[Configurator](#)   [+ Add rule manually](#)

#### Step 4. Connectivity protocols settings

You must configure one or more protocols for connecting the application to Blitz Identity Provider.

**Protocols**

**SAML**   OAuth 2.0   Simple   REST

SAML protocol is not configured. [Configure](#)

The following connectivity protocols are supported:

- SAML - for connecting applications via SAML 1.0, 1.1, 2.0, and WS-Federation for user identification and authentication.
- OAuth 2.0 - for connecting applications via OAuth 2.0, OpenID Connect 1.0 (OIDC) for user identification and authentication. Dynamic client registration can be configured within this protocol.
- Simple - for connecting web applications to perform identification and authentication by substituting a login and password from a proxy server into the application, if the application does not support SAML/OIDC connectivity.
- REST – connecting applications that use the REST services of Blitz Identity Provider for account registration/ modification, user authentication device management.
- RADIUS – to connect to network services using the RADIUS protocol.

If an organization plans to develop or modify its own applications to connect them to Blitz Identity Provider, developers should review [Integration Guide](#) (page 294).

If an organization plans to connect applications that have native support for SAML 1.0, SAML 1.1, SAML 2.0, WS-Federation, or OIDC (OpenID Connect 1.0, OAuth 2.0) connectivity to Blitz Identity Provider, the following subsections describe the general settings on the Blitz Identity Provider side of connecting an arbitrary SAML/OIDC-enabled application.

## 2.3.2 Operation schemes of SSO technologies

This section provides operation schemes of common single sign-on technologies such as OAuth 2.0 and SAML.

### Connecting a web app via OIDC

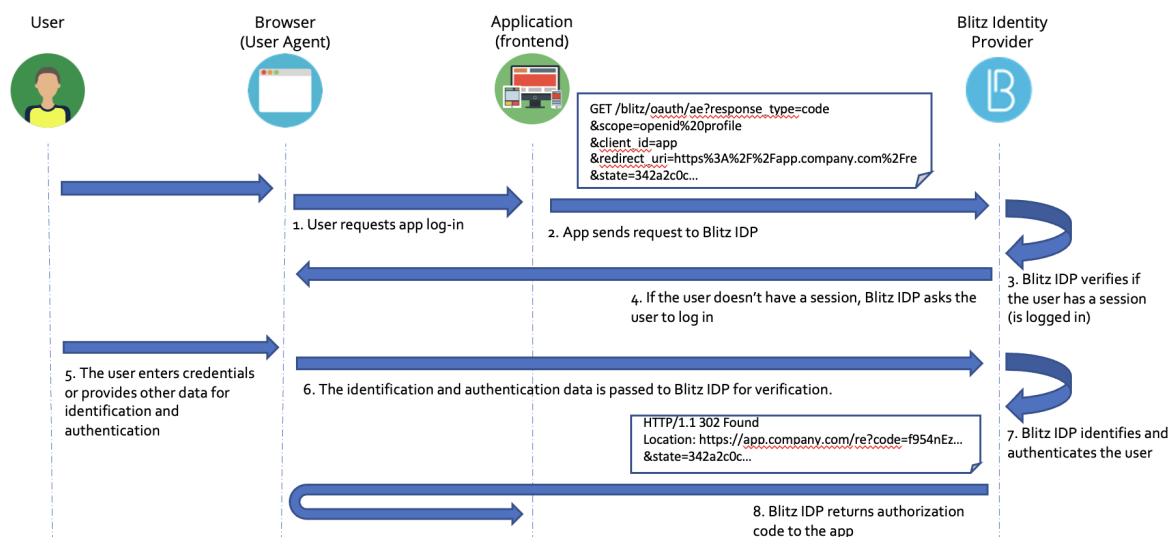
The interaction of the web application with Blitz Identity Provider by OIDC includes the following steps:

**Note:** This process coincides with the Authorization Code Grant application authorization model provided for in the OAuth 2.0 specification.

1. The application sends a request for user identification and authentication via a web browser to the Blitz Identity Provider address.
2. Blitz Identity Provider identifies/authenticates the user.
3. Blitz Identity Provider receives the user's consent to transfer information about him to the application (for applications hosted on the domain `company.com`, consent is provided automatically without the user's request).
4. Blitz Identity Provider redirects the user back to the application via the web browser and transmits the authorization code to the application.
5. The application uses the authorization code to generate a request for an identification token, an update token, and an access token.
6. The application receives a response containing the necessary tokens.
7. The application requests user data using an access token. If necessary, the application can verify the identification token and extract the user ID and additional attributes from this token.

The figures show the processes of obtaining an authorization code, tokens, and user data.

### Getting the authorization code:



## Getting security tokens:

Application  
(backend)Blitz Identity  
Provider

```
POST /blitz/oauth/te
Content-Type: application/x-www-form-urlencoded
Authorization: Basic bG9jYWxo...

grant_type=authorization_code
&code=f954nEz...
&redirect_uri=https%3A%2F%2Fapp.company.com%2Fre
```

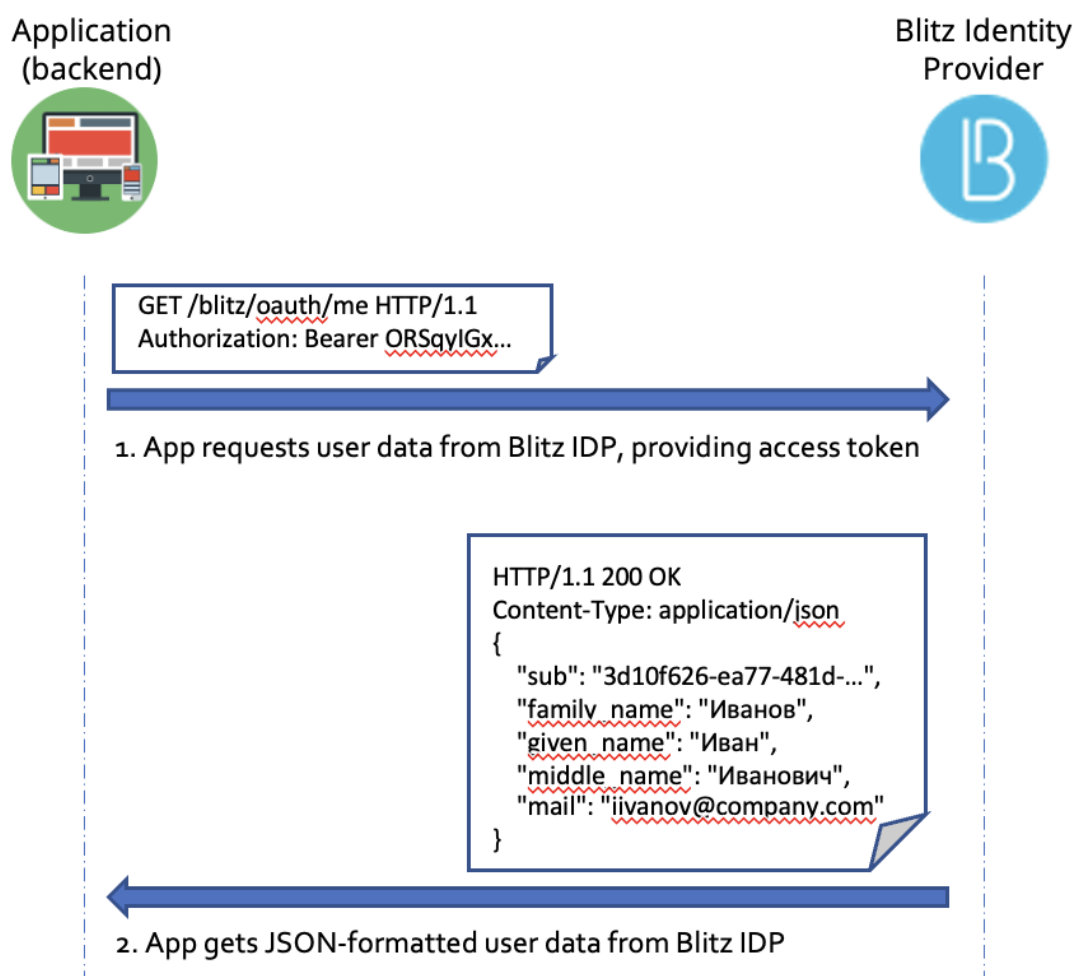
1. Application requests security tokens from Blitz IDP

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "access_token": "ORSqylGx...",
  "token_type": "bearer",
  "expires_in": 3600,
  "refresh_token": "FepnjwwA...",
  "id_token": "evJhbGci..."
}
```

2. Application receives access token, id token, refresh token



## Getting user data:



## Connecting a mobile app via OIDC

The interaction of the mobile application with Blitz Identity Provider in addition to the standard tools of the OIDC/OAuth 2.0 protocol uses the specifications:

- [RFC 7591 OAuth 2.0 Dynamic Client Registration Protocol](https://tools.ietf.org/html/rfc7591)<sup>38</sup>,
- [RFC 7592 OAuth 2.0 Dynamic Client Registration Management Protocol](https://tools.ietf.org/html/rfc7592)<sup>39</sup>.

The interaction of the mobile application with Blitz Identity Provider includes the following steps:

1. Dynamic registration of a mobile application instance in Blitz Identity Provider. Getting an application instance from Blitz Identity Provider a unique `client_id`/`client_secret` pair.
2. The user's initial login to the mobile application using Blitz Identity Provider. The user sets a PIN code or Touch ID/Face ID. Saving the encrypted `client_id`/`client_secret` pair received from Blitz Identity Provider on the device.
3. Secondary user inputs using a PIN or Touch ID/Face ID. Authorization in Blitz Identity Provider using the encrypted `client_id`/`client_secret` pair.

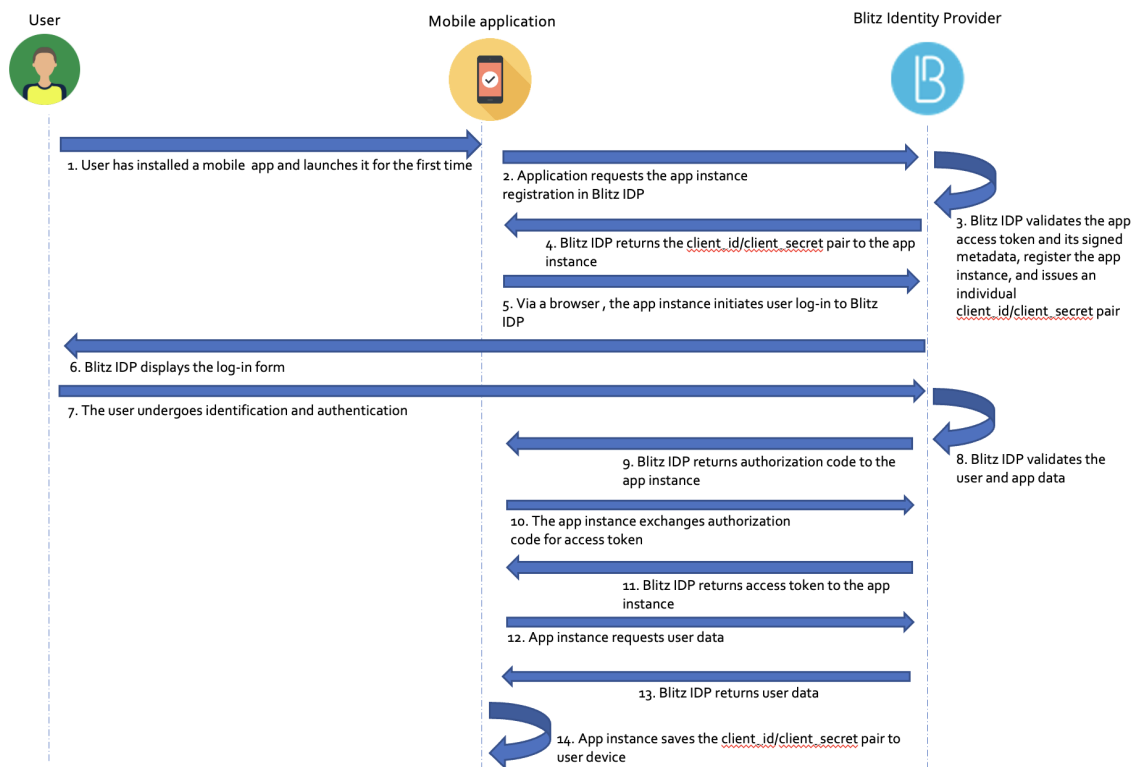
<sup>38</sup> <https://tools.ietf.org/html/rfc7591>

<sup>39</sup> <https://tools.ietf.org/html/rfc7592>

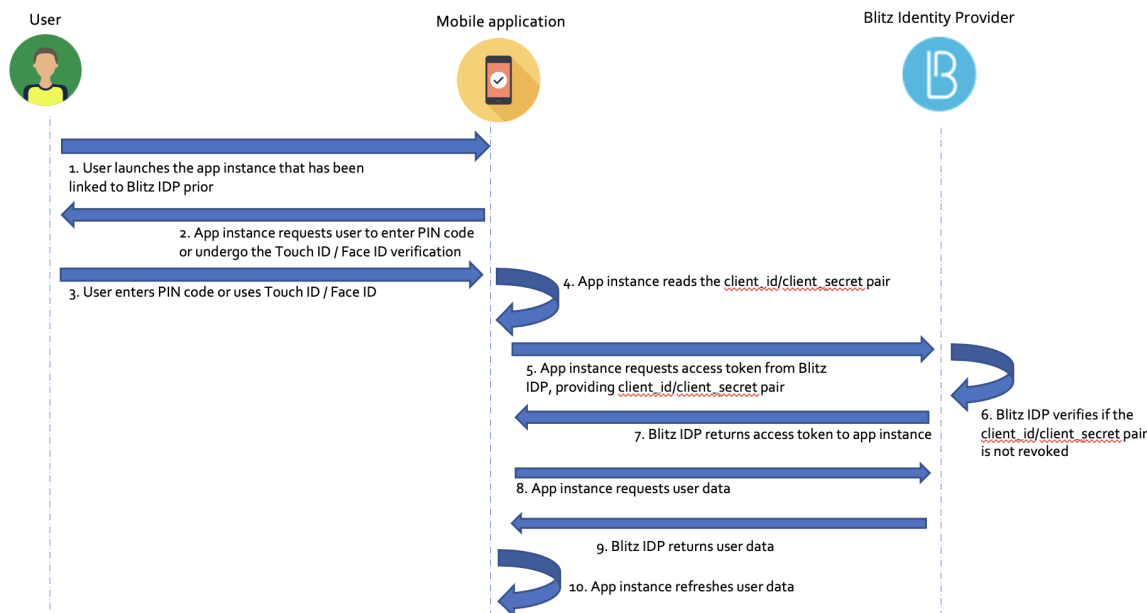
- Deleting the received `client_id` / `client_secret` pair in Blitz Identity Provider during the user's logout (account change, account logout) from the mobile application.

Schematically, the sequence of actions of stages 1-2 is shown in the first figure, and stage 3 is shown in the second.

The user's first login to the mobile application:



### Repeated user logins to the mobile application:



### Connecting an app via SAML

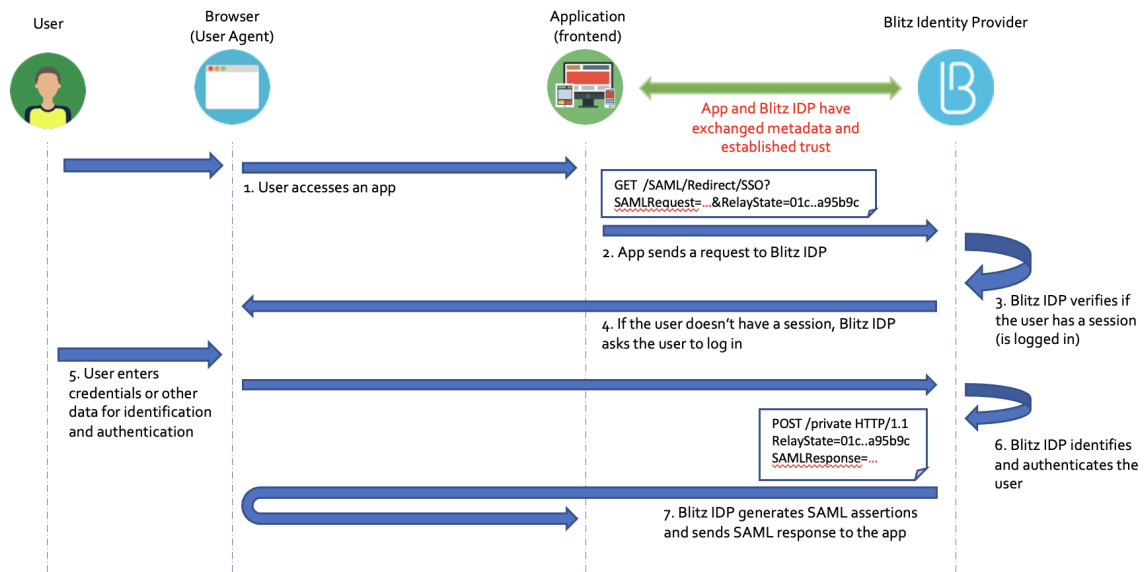
During the interaction, the application (service provider) sends a SAML request to Blitz Identity Provider for user identification (SAML Request). The request is an XML document designed in accordance with the SAML standard. The request contains the identifier of the application requesting identification, called the `entityID`, as well as additional service information. The request itself is transmitted electronically signed by the application. The HTTPS protocol is used as the transport protocol for transmitting a message, and the identification provider is called via HTTP Redirect. This means that the request from the application to Blitz Identity Provider is made indirectly, through the user's browser, and direct network interaction between the application and Blitz Identity Provider is not required when using SAML.

After receiving a SAML identification request, Blitz Identity Provider identifies the request belonging to a specific application, after which it displays a single sign-on web page to the user to identify and authenticate the user. In case of successful identification and authentication of the user, Blitz Identity Provider transmits a SAML Response to the application (service provider). Depending on the set interaction settings, the request can be signed and encrypted. XML Signature and XML Encryption standards are used for signature generation and encryption. The HTTPS protocol is used as a transport protocol for transmitting a message with identification results, and the service provider is called via HTTP POST.

After receiving a SAML response from Blitz Identity Provider, the application verifies its signature, performs decryption, and then extracts user identification data (identifiers, attributes, permissions) from SAML statements.

The process of interaction between the application and Blitz Identity Provider using SAML is shown in the figure.

## User identification using SAML



### 2.3.3 Configuring SAML and WS-Federation

#### Connection via SAML 1.0/1.1/2.0

When connecting an application via SAML, you must make the following settings:

- [load the SAML metadata of the application to be connected](#) (page 169);
- make sure that the SAML Profile switch is set to SAML 2.0 Web SSO Profile;
- in the SAML profile block, click Configure. In the fields that appear, specify:
  - specify whether to sign SAML attributes (`SAML Assertions`) in Blitz Identity Provider responses;
  - specify whether to encrypt SAML-attributes in Blitz Identity Provider responses;
  - specify whether to encrypt SAML identifiers (`SAML NameIds`) in Blitz Identity Provider responses;
  - specify whether to include a list of assertions with attributes in Blitz Identity Provider responses;
- specify which SAML user attributes from Blitz Identity Provider to pass to the application. SAML attributes must be [pre-configured](#) (page 170) in the SAML section of the Admin Console.

Protocols

SAML OAuth 2.0 Simple REST

Metadata

```

1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
3   xmlns:esia="urn:esia:shibboleth:2.0:mdext"
4   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
5   entityID="https://lab-app.reaxoft.loc/owa/"
6   <md:SPSSODescriptor protocolSupportEnumeration="urn:mace:shibboleth:wsf-prp:1.0:protocol">
7     <md:AssertionConsumerService Binding="urn:mace:shibboleth:1.0:bindings:HTTP-POST-wsigin"
8       Location="https://lab-app.reaxoft.loc/owa/"
9       index="1"/>
10  </md:SPSSODescriptor>
11 </md:EntityDescriptor>

```

---

SAML profile SAML 2.0 Web SSO Profile WS-Federation Passive Requestor Profile

Sign assertions    
 A rule to sign assertions

Encrypt assertions    
 A rule to encrypt assertions

Encrypt NameIds    
 A rule to encrypt NameIds

Send user SAML-assertions in a Attribute Statement block

---

User attributes   
 Define attributes and their SAML-specific names to send to the application

SAML-attribute	Send	
transientid	<input type="checkbox"/>	<input type="button" value="✖"/>
upn	<input checked="" type="checkbox"/>	<input type="button" value="✖"/>

[+ Add](#)

## Connection via WS-Federation

When connecting an application via WS-Federation, the following settings must be configured:

- [load the SAML metadata of the application to be connected](#) (page 169);
- set the SAML profile switch to WS-Federation Passive Requestor Profile;
- in the SAML profile block, click Configure. In the fields that appear, specify:
  - specify whether to sign assertions (`Assertions`) in Blitz Identity Provider responses;
  - specify the lifetime of assertions in the response. ISO 8601 format should be used to specify the duration of the `period`<sup>40</sup>, e.g. PT5M - 5 minutes;
  - specify whether to include a list of assertions with attributes in Blitz Identity Provider responses;
- specify which user attributes from Blitz Identity Provider to pass to the application. Attributes must be [pre-configured](#) (page 170) in the SAML section of the Admin Console.

<sup>40</sup> <http://www.ifap.ru/library/gost/86012001.pdf>

## Uploading SAML metadata

You can use either method to upload SAML metadata of an application:

- To upload a ready-made XML file, click Open file.

- To use the metadata builder, click Generate metadata. Enter the following data:
  - The assertion handler service URL (`AssertionConsumerService`),
  - The single logout service URL (`SingleLogoutService`),

- Signature certificate,
- Encryption certificate.

Protocols

SAML OAuth 2.0 Simple REST RADIUS

Set parameters for metadata generation

URL AssertionConsumerService

URL SingleLogoutService

Signature certificate

Encryption certificate

Generate Cancel

Click Generate. As a result, the metadata file will be automatically generated based on the entered data.

### Configuring SAML attribute

The SAML section of the Admin Console is used to register user SAML attributes in Blitz Identity Provider.

To add a new SAML-attribute you must:

1. Click on the Add a new SAML attribute link.
2. Enter:
  - name of the SAML-attribute (this is what will be displayed when connecting SAML applications);
  - attribute source (all attributes defined in the Data sources section are displayed).
3. Press Add. The attribute will be added.
4. Define attribute encoders. This requires:
  - click on the link Add encoder;
  - choose the type of encoder; it should be noted that the type of encoder depends on the protocol version the service provider (connected application) works with;
  - name of the SAML attribute that will be sent to the service provider (within this encoder type);
  - a short name to be given to the service provider (within this encoder type);
  - name format.

If necessary, multiple encoders of the selected SAML attribute can be defined (each encoder must belong to a different encoder type).

### Attributes

Define attributes from the store that can be sent to SAML-applications (Service Providers) as SAML-attributes

Find

urn:blitz:username
urn:blitz:name
urn:blitz:surname
urn:blitz:mail

[+ Add a new SAML-attribute](#)

SAML-attribute properties

Identifier

Source

Encoder

Type

Name

Short name

Name format

[+ Add an encoder](#)

Delete SAML-attribute

## 2.3.4 OAuth 2.0 and OpenID Connect 1.0

### Configuring the application

When connecting an application via OAuth 2.0 or OpenID Connect 1.0 (OIDC), you must set the following application interaction settings in Application interaction settings block:

- specify the secret key (or use the default generated key) of the connected application (`client_secret`) to be used by the connected application when accessing the Blitz Identity Provider (if not specified, the client application must be authenticated otherwise, for example, using a TLS proxy);
- specify an additional secret key (`client_secret`) for the connected application. It is recommended for cases when it is necessary to provide smooth change of `client_secret` for this application;
- specify a predefined return link (`redirect_uri`) - the URL to which the user will be redirected by default after authorization (`redirect_uri`);
- specify valid return link prefixes - the prefix is used to validate return links (`redirect_uri`) passed in authentication requests from applications. If a return link is specified in an authentication request and it does not match any of the specified prefixes, authentication will be denied;
- allowable permissions - the permissions (`scope`) that this application is allowed to request;

**Note:** You can [configure](#) (page 109) Blitz Identity Provider to store user access tokens from external identity providers. If the application needs to [receive](#) (page 385) stored access tokens via REST API, select the following system permissions for it: `fed_tkn_any` (all external providers) or `fed_tkn_{$fedPointType}_{$fedPointName}` (external provider with the `{$fedPoint-`




`Type` type and `fedPointName` name). These permissions must be preliminarily set in the general OAuth settings on the OAuth tab.


- default permissions - the permissions (`scope`) that will be granted to the application by default after authentication. If not specified, the required permissions must always be explicitly stated in the authentication request;
- check the option *“Do not require the user to agree to provide access to data about him/herself”* if necessary. If this option is checked, the consent page will not be displayed when the user logs in for the first time;
- check the *“Mandatory use of Proof Key for Code Exchange (RFC 7636) for Authorization code grant type”* option, if authentication requests must be validated according to RFC 7636;
- select, if necessary, the authentication method when accessing the token service. The specified authentication methods must be used when accessing the token service (`token endpoint`). If empty, all methods are available;
- select valid `grant types` if necessary. The parameter specifies the list of `grant types` that will be available to the application. If the list is empty, all `grant types` are available;
- select valid `response types` if necessary. The parameter specifies the list of `response types` that will be available to the application when accessing the authorization URL (`authorization endpoint`). If the list is empty, all `response types` are available;
- specify the lifetime of the access token (in seconds). If the parameter is not specified, it is taken from the general settings in the *“OAuth 2.0”* section;
- specify the default mode of issuing access tokens. Blitz Identity Provider provides two modes of issuing access tokens (`access_token`):
  - offline mode - when requesting an access token, a perpetual refresh token (`refresh_token`) will also be issued, which can be used to obtain a new access token. It is recommended that an application use this mode if it needs to retrieve up-to-date user data from Blitz Identity Provider outside of the validity time of the user session. For example, if an application is doing a mailing and wants to get an up-to-date email address from Blitz Identity Provider before sending it.
  - online mode - only the access token will be issued. The application is recommended to use this mode if it is sufficient to receive actual user data at the moment of login (during the active user session).

The mode of issuing access tokens may be explicitly specified in the authentication request; if not specified, the default mode is used.

- specify the lifetime of the update token (in seconds). If the parameter is not specified, it is taken from the general settings in the *“OAuth 2.0”* section;
- specify the assertions to be added to the identity token (`id_token`). If the application communicates with Blitz Identity Provider using the OIDC protocol (OpenID Connect 1.0), you must also specify `openid` as one of the authorizations (`scope`). Then in exchange for the authorization code, when calling `Token Endpoint`, not only an access token (`access token`) and a refresh token (`refresh token`), but also an identification token (`id_token`) will be issued. The identity token will include the user identifier `sub` as well as the additional attributes listed in this setting. It is possible to add both the attributes configured in *“Data sources”* and additional attributes (see [Adding attributes to an identity token](#) (page 177) for details);
- select access token format - you can choose `opaque` or `JWT`. If the parameter is not specified, it is taken from the general settings in the *“OAuth 2.0”* section.

**Interaction settings**

**Secret (client\_secret)**    
 Application's secret (client\_secret). If defined, this secret should be used by the application when making a request to Blitz Identity Provider

**Extra secret (client\_secret)**    
 Application's extra secret (client\_secret). If defined, this extra secret should be used by the application when making a request to Blitz Identity Provider

**Predefined redirect uri (redirect\_uri)**   
 URL used for user redirection after successful authorization (redirect\_uri)


**Redirect uri prefixes**   
 A prefix is used to check the redirect uri. If the authorization request includes an redirect uri that doesn't correspond to any prefix, then the authentication is rejected

**Available scopes**   
 The scopes that will be available to the application.

**Default scopes**   
 Scopes that are granted by default after authorization. If there are no default scopes, then the request must explicitly include the required scopes.

Do not require user consent to the provision of access to his data


Mandatory use of Proof Key for Code Exchange (RFC 7636) for Authorization code grant type

**Authentication method to access token issuance service**    
 The specified authentication method should be used when accessing token issuance service (token endpoint). If the value is empty, all methods are available

**Valid grant type**    
 List of grant type that will be available to the application. If the list is empty, all grant type are available

**Valid response type**   
 List of response type that will be available to the application instance when accessing the authorization URL (authorization endpoint). If the list is empty, all response type are available.

**Access token lifetime**   
 Specify the number of seconds before the access token will be invalid. If not specified, then it is taken from general settings.

**Default mode of issuing access tokens**    
 Mode of issuing access tokens, if not explicitly specified in the request. In online mode the refresh token is not issued

When using the `logout`<sup>41</sup> function in an application, the following settings must be specified in the “*Exiting the application*” block:

- specify prefixes of return links on exit. You must list the prefixes of valid URLs of user redirect pages after the application initiates a logout. You can specify one or more return link prefixes;
- predefined exit return link - the link to which the user will be redirected after logout from the application,

<sup>41</sup> [https://openid.net/specs/openid-connect-rpinitiated-1\\_0.html#RPLLogout](https://openid.net/specs/openid-connect-rpinitiated-1_0.html#RPLLogout)

if the `post_logout_redirect_uri` return address was not passed in the parameters of the logout call from the application;

- check the option “Do not show the logout confirmation screen to the user” - if this setting is not checked, the user will be shown a screen requesting logout confirmation;
- link to clear user session in browser (`Front channel`) - the specified application handler address will be called from browser frame in case of user logout initiation;
- check the option “Add the session ID and issuer to the session cleanup link in the browser (`Front channel`)” if necessary - in this case the session identifier (`id`) will be passed to the browser application logout handler;
- link to the user’s session cleanup in the application (`Back channel`) - the specified application handler address will be called from the Blitz Identity Provider server if a user logout is initiated;
- check the option “Add the session ID and issuer to the session cleanup link in the application (`Back channel`)” if necessary - in this case `logout_token` containing user session identifier (`sid`) will be sent to the application handler address called from Blitz Identity Provider server in case of user logout initiation.

**Logout**

**Logout redirect uri prefixes**   
 A prefix is used to check the redirect uri. If the logout request includes an `post_logout_redirect_uri` that doesn't correspond to any prefix, then the logout is rejected

**Predefined logout uri**   
 URL used for user redirection after successful logout

Do not show the user the logout confirmation screen

**Link for cleaning up a user session in the browser (`Front channel`)**   
 URL to which the browser will be redirected to clean up session information

Add session ID and issuer to the session cleanup link in the browser (`Front channel`)

**Link for cleaning up a user session in the application (`Back channel`)**   
 URL to which a request will be made to clean up session information

Add session ID and issuer to the session cleanup link in the application (`Back channel`)

When an application uses the [Device Authorization Grant](https://tools.ietf.org/html/rfc8628)<sup>42</sup> specification (for example, to connect IOT devices, smart TV, chat bots, voice assistant applications) in the Application interaction settings block in the `Acceptable response type` parameter, add the `device code` option, and in the `Allowed grant type` parameter, add the `urn:iETF:params:oauth:grant-type:device_code` option. Also in the Device Authorization Grant block you should set the following settings:

- format of the user’s code, for this you should use regular expressions;
- user code lifetime;
- link to the custom code entry page;
- check the option “Add user code to URLs” if necessary. In this case Blitz Identity Provider will return not only a link to the user code input page (for example, `https://test.ru/device`), but also a link with the code as a parameter (for example, `https://test.ru/device?uc=676-267-324`).

<sup>42</sup> <https://tools.ietf.org/html/rfc8628>

## Device Authorization Grant

User's code format	<input type="text" value="[0-9]{3,3}-[0-9]{3,3}-[0-9]{3,3}"/>
	The format is specified as a template based on a regular expression that generates the user's code to bind the device. Example: [0-9]{2,3}-[0-9]{2,3}
User's code lifetime	<input type="text"/>
	Specify the number of seconds before the user's code will be invalid. If not specified, then it is taken from general settings.
Link to the user's code entry page	<input type="text"/>
	If the link is not specified, it is generated automatically
	<input type="checkbox"/> Добавлять в URL пользовательский код

## General OAuth 2.0 settings

The “*OAuth 2.0*” section of the Management Console is used to specify general OAuth 2.0 settings, as well as to configure a set of permissions (*scope*).

Properties

URL with Blitz Identity Provider metadata	<input type="text" value="/blitz/oauth/.well-known/openid-configuration"/>	When connecting applications using OpenID Connect in the settings of these applications you may need to specify this link to the metadata file
URL for authorization	<input type="text" value="/blitz/oauth/ae"/>	The request for authorization should be sent to this URL (authorization endpoint)
URL to get and refresh a token	<input type="text" value="/blitz/oauth/te"/>	The request for getting and refreshing an access token should be sent to this URL (token endpoint)
Access token lifetime, seconds	<input type="text" value="3600"/>	
Access token format	<input type="text" value="jwt"/>	
Refresh token lifetime, sec	<input type="text" value="31536000"/>	

Client authentication using proxy TLS. To use proxy TLS to authenticate clients the bilateral TLS-connection should be established. In the certificate's Common Name (CN) field the client's domain should be indicated

In the “*OAuth 2.0*” section of the Admin Console, you can view the various Blitz Identity Provider handler URLs associated with OAuth 2.0 and OIDC:

- “URL with Blitz Identity Provider metadata” - this link contains dynamically updated Blitz Identity Provider settings (metadata) ([specification](#)<sup>43</sup>). Application developers do not need to specify all of the following URLs in their application configuration, but can use a single link to this metadata in the settings;
- “URL for authorization” is the address of the OAuth 2.0 Authorization Endpoint handler for requests through the browser for an authorization code;
- “URL to get and refresh a token” - the address of the OAuth 2.0 Token Endpoint handler to retrieve security tokens (*access\_token*, *id\_token*, *refresh\_token*).

If necessary, you can:

- change the “*Access token lifetime*” used by default when issuing tokens for all applications;
- specify the “*Access token format*” used by default when issuing tokens for all applications: string (*opaque*) or JWT;

<sup>43</sup> <https://tools.ietf.org/html/draft-ietf-oauth-discovery-10>

- change the “*Update marker lifetime*” used by default when releasing tokens for all applications;
- check the “*Authentication of client systems using Proxy TLS*” option. In this case, applications must be configured to communicate with Blitz Identity Provider via a proxy server and a two-way TSL connection must be established. The “*Common Name (CN)*” field of the system certificate must contain the system domain of the connected application.

Under “*Device Authorization Grant*” you can define the general settings for interaction with applications according to the Device Authorization Grant specification. Here you can specify:

- lifetime of the user’s code (in seconds);
- is the minimum allowed interval for polling the device binding code status in seconds. If the application polls the Blitz Identity Provider service more often than specified in this parameter, an error will be returned.

If necessary, you can specify different settings related to the Device Authorization Grant specification for each application.

#### Device Authorization Grant

Users code lifetime, sec	<input type="text" value="300"/>
Allowed interval of status poll of the device binding status code in seconds	<input type="text" value="5"/>

For correct operation of interaction with applications using the OAuth 2.0 protocol, it is necessary to define permissions (*scope*). To do this, you need to specify:

- scope name;
- scope description (it will be displayed to the user on the consent page);
- attributes of the user that will be provided under this permission (attributes must be defined in the “*Data sources*” menu);
- whether the scope is system permissions - such permissions are only granted to applications using OAuth 2.0 Client Credentials Flow (not in the context of individual user permissions, but general ones).

Available scopes

Specify scopes that can be requested by applications (clients). If necessary, specify the user attributes that can be obtained by those scopes

Scope name	Description	User attributes	Client	
<input type="text" value="openid"/>	<input type="text" value="User identification"/>	<input type="text"/>	<input type="checkbox"/>	<input type="button" value="✖"/>
<input type="text" value="profile"/>	<input type="text" value="Profile data"/>	<input type="text" value="✖ surname ✖ mail ✖ mobile ✖ uid ✖ name ✖ email ✖ username"/>	<input type="checkbox"/>	<input type="button" value="✖"/>
<input type="text" value="email"/>	<input type="text" value="mail"/>	<input type="text" value="✖ mail ✖ email"/>	<input type="checkbox"/>	<input type="button" value="✖"/>

[+ Add a scope](#)

**Attention:** For OpenID Connect 1.0 authentication to work correctly, you must ensure that a permission named `openid` is defined in this section of the console. You can also specify the attributes to be passed with this permission. In this case, the attributes are retrieved by using an access token issued for the `openid` permission.

**Important:** You can [configure](#) (page 109) Blitz Identity Provider to store user access tokens from external identity providers. If applications connected over OAuth 2.0 need to [receive](#) (page 385) stored access tokens via REST API, specify the following system permissions in this console settings block: `fed_tkn_any` (all external providers) or `fed_tkn_${fedPointType}_${fedPointName}` (external provider with the `${fedPointType}` type and `${fedPointName}` name). These permissions must also be specified in the application-specific OAuth settings.

### Adding attributes to an identity token

Applications connected using the OpenID Connect 1.0 protocol can receive data in the identity token. The list of attributes to be passed in the identity token must be specified in the `Added to identity token (id_token) assertions` clause of the protocol settings.

In addition to the stored attributes, assertions can be added to the identity token:

- received when the user logs in by electronic signature. This may be data on the electronic signature key certificate, data on the physical/legal person from the certificate;
- defined in the authentication flow.

To obtain assertions from the electronic signature key certificate, the `blitz.conf` configuration file must be edited by adding the following structure to the `blitz.prod.local.idp.login.methods.x509` configuration block:

```
"claims" : [
  {
    "name" : "attr_name",
    "value" : "cert_attr_name"
  }
],
```

In this structure, `attr_name` is the name of the attribute to be used in the identification token, and `cert_attr_name` is the attribute designation in the certificate (examples of available values are given in the table).

### Example of data obtained from electronic signature key certificate

Attribute designation in certificate	Description
SUBJECT.OGRN	OGRN of the organization
SUBJECT.OGRNIP	OGRNIP of an individual entrepreneur
SUBJECT.INN	TIN of the organization
SUBJECT.E	Company email of official
SUBJECT.O	Organization name
SUBJECT.ST	Organization region
SUBJECT.L	Organization location
SUBJECT.STREET	Street, house, office number of the organization
SUBJECT.O	Division of the official
SUBJECT.T	Position of the representative
SUBJECT.<OID>	A value from an attribute with the specified OID. For example, SUBJECT.1.2.643.100.5 allows an attribute with OI D 1.2.643.100.5 to be accessed

An example of the structure added to the configuration file:

```
"claims" : [
  {
    "name" : "org_OGRN",
    "value" : "SUBJECT.OGRN"
  },
  {
    "name" : "org_INN",
    "value" : "SUBJECT.INN"
  },
  {
    "name" : "org_email",
    "value" : "SUBJECT.E"
  },
  {
    "name" : "org_name",
    "value" : "SUBJECT.O"
  }
],
```

To be able to define session assertions in the login procedure, the corresponding assertions must also be defined in the configuration file. For this purpose, the `blitz.prod.local.idp.login` section of the configuration file must have the `sessionClaims` attribute added with a list of assertions that can be defined in the procedure.

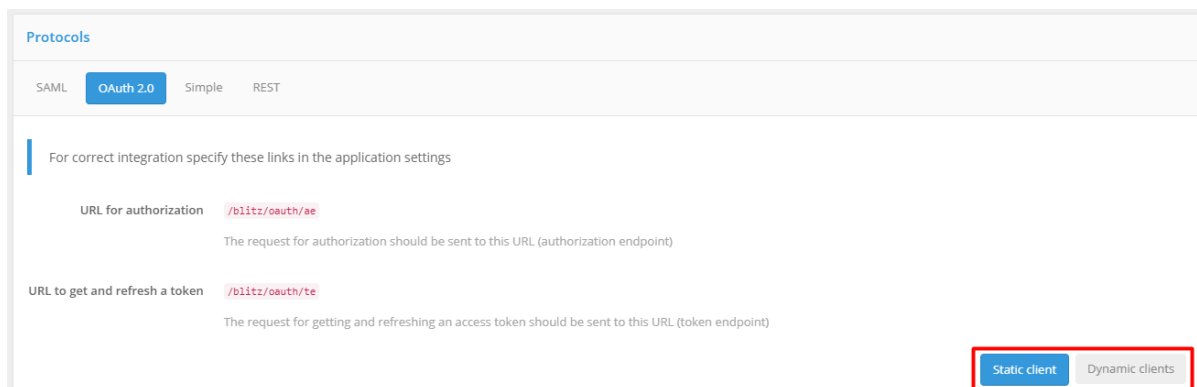
For example, the following entry allows you to define the `custom_attr` attribute:

```
"sessionClaims" : [
  "custom_attr"
]
```

## Configuring Dynamic OAuth 2.0 Client Registration

To enable dynamic client registration, you must take the following steps:

- register the application and configure the OAuth 2.0 connection protocol for it according to the documentation (see [General OAuth 2.0 settings](#) (page 175));
- in the OAuth 2.0 settings for this application, click the “Dynamic clients” tab.



Specify the basic settings for dynamic client registration:

- allow dynamic client registration;
- specify the assertions that may be directly transferable. These assertions may be specified in the application instance registration request. If they are present in the application metadata (`software_statement`), the value from the metadata will be prioritized. It is recommended to allow only the device type (`device_type`) to be passed.

Create primary tokens for the application. Primary tokens are used to authorize application instances when they are registered.

Generate application metadata (`software_statement`). This metadata is passed as a assertion in the application instance registration request. You can specify as metadata attributes:

- application version (mandatory attribute). The application version must match the version of the primary token used by the application;
- return link prefixes. The prefix is used to validate return links (`redirect_uri`). If a return link is specified in an authentication request and it does not match any of the specified prefixes, authentication will be denied;
- allowable permissions - the permissions (`scope`) that will be available to the application;
- authentication method when accessing the token service. The specified authentication method must be used by the application instance when accessing the token service (`token endpoint`);
- permissible values of `grant type`. A list of `grant type` that will be available to the application instance;
- permissible values of `response type`. A list of `response type` that will be available to the application instance when accessing the authorization URL (`authorization endpoint`)

Note that the specified metadata attributes must match the OAuth 2.0 parameters defined for the application (“Static client”).

Once the application metadata is signed, it should be passed along with the primary tokens to the developers of the plug-in application.

An example of dynamic client registration settings is shown in the figure below.



### Protocols

SAML
OAuth 2.0
Simple
REST

For correct integration specify these links in the application settings

**URL for authorization** /blitz/oauth/ae  
The request for authorization should be sent to this URL (authorization endpoint)

**URL to get and refresh a token** /blitz/oauth/te  
The request for getting and refreshing an access token should be sent to this URL (token endpoint)

Static client
Dynamic clients

---

#### Dynamic client registration settings

**Allow dynamic client registration**

**Software ID (software\_id)** headless-app  
Used for dynamic client registration

**The assertions allowed for direct transfer** \*Device type (device\_type)  
These assertions are allowed to be specified in the request to register an application instance

Change

---

#### Signing application metadata

Sign application metadata (software\_statement). This metadata is passed as an assertion in the request to register an application instance

**Software version**   
The software version in the metadata must correspond to the version of the primary token

**Redirect URI prefixes** To add a new prefix, enter it and press Enter  
The prefix is used to verify redirect URI (redirect\_uri). If a redirect URI is specified in an authentication request and does not match any of the specified prefixes, authentication will be rejected

**Allowable scopes** To add a new scope, enter it and press Enter  
Scopes (scope), which will be available to the application.

**Authentication method to access token issuance service** ▼  
The specified authentication method should be used by application instance when accessing token issuance service (token endpoint)

**Valid grant type values** To add a new grant type, enter it and press Enter  
List of grant types that will be available to the application instance

**Valid response type values** To add a new response type, enter it and press Enter  
List of response type that will be available to the application instance when accessing the authorization URL (authorization endpoint)

Generate

---

#### Primary tokens

Primary tokens are used to authorize application instances during their registration

Primary tokens not found

Software version 
Issue

### 2.3.5 Simple

You can use this method to connect an application to Blitz Identity Provider under the following conditions:

- An application cannot be connected to Blitz Identity Provider using standard SAML or OIDC protocols.
- The application is a web application deployed in its own infrastructure (On-Premise). User access to applications can be organized through a reverse proxy server.

To connect an application to Blitz Identity Provider using the Simple protocol, you must:

1. In the application settings in the Admin Console, select the Simple protocol and set its settings:
  - SSL - a setting that specifies whether the proxy calls the application connected by Simple via HTTP or HTTPS. It is recommended to use an existing web server of the application as a proxy server protecting the application, in which case the connection between the proxy server and the application will be made without TLS/SSL encryption.
  - Form selector - specifies a CSS selector to define the position of the login form on the plug-in application page.
  - Login field selector - specifies a CSS selector to define the position of the login field on the login page of the plug-in application.
  - Default Logout URL (optional setting) - specifies which URL Blitz Identity Provider should call when it is necessary to initiate a logout in a Simple connected application in the case of a single logout in Blitz Identity Provider.
  - URL to go to after a successful logout - specifies which URL Blitz Identity Provider should call to redirect the user after a successful logout initiated by a Simple connected application.
  - JavaScript (optional setting) - JS code embedded in the login page of the Simple plug-in application, which allows to process the response received from the application with login results (check that the login was successful) and show an error about it in Blitz Identity Provider.

Example value:

```
var fm = document.querySelector('form[name=login]');

if (fm) {
  document.body.style.display = "none";
  var err = document.getElementById('lost-password');
  var errKey = err && err.innerHTML.indexOf('Incorrect password.') !== -1 ?
  → 'incorrect_password' : 'unknown_error';
  var kvp = document.location.search.substr(1).split('&');
  kvp.push([encodeURIComponent('error'), encodeURIComponent(errKey)].join('='));
  window.location.search = kvp.join('&');
}

var aLogout = document.querySelector('#logout');
var href = aLogout ? aLogout.getAttribute("href") : null;
if (href) {
  var lp = encodeURIComponent(href);
  var slp = document.createElement('script');
  slp.setAttribute('src', 'https://idp.company.com/blitz/simple/slp?app=app_
  → id&lp=' + lp);
  document.head.appendChild(slp);
}
```

An example of Simple protocol settings for an application is shown in the figure below.

**Protocols**

SAML   OAuth 2.0   **Simple**   REST

**SSL**

**Form selector**   
CSS-selector is used to detect the login form on the page

**Username field selector**   
CSS-selector is used to detect the username field on the page

**Default logout URL**   
URL for redirection to make a logout

**URL for redirection after a successful logout**   
URL for redirection after a successful logout from the application

**JavaScript**

**Save**

2. Set the settings for proxying requests to the application on the web server.

The example of configuration file for nginx web-server:

```
map "" $idp_host {
    default <server hostname>:9000;
}

map "$http_Blitz_Idp" $idp_post_login {
    default "0";
    "prepare-login" "1";
}

map "$arg_passive" $activLogout {
    default "1";
    "true" "0";
}

upstream oc-web {
    server <application server hostname>:<application port>;
}

server {
    listen 80;
    server_name <application domain name>;
    # enforce https
    return 301 https://$server_name$request_uri;
}
```

(continues on next page)

(continued from previous page)

```

server {
    listen          443 ssl;
    server_name     <application domain name>;

    resolver        172.27.0.20 172.25.0.50 valid=300s;
    #resolver       8.8.8.8 valid=300s;

    #ssl_certificate /etc/nginx/cert/<path to SSL certificate>.pem;
    #ssl_certificate_key /etc/nginx/cert/<path to SSL certificate key>.pem;

    #ssl_certificate /etc/letsencrypt/live/app.company.com/fullchain.pem; #_
    ↪managed by Certbot
    #ssl_certificate_key /etc/letsencrypt/live/app.company.com/privkey.pem; #_
    ↪managed by Certbot

    access_log      /var/log/nginx/oc-acs.log full;
    error_log       /var/log/nginx/oc-err.log error;

    ### force timeouts if one of backend is died ##
    proxy_next_upstream error timeout invalid_header http_500 http_502 http_
    ↪503 http_504;

    ### Set headers #####
    proxy_set_header    Accept-Encoding    "";
    proxy_set_header    Host               $host;
    proxy_set_header    X-Real-IP         $remote_addr;
    proxy_set_header    X-Forwarded-For   $proxy_add_x_forwarded_for;
    proxy_set_header    X-Forwarded-Proto $scheme;
    add_header          Front-End-Https   on;
    proxy_redirect      off;

    proxy_set_header    Cookie "$http_cookie;domain2auth=$host";
    proxy_hide_header   Content-Security-Policy;

    add_header Content-Security-Policy "default-src 'self' https://$idp_host;_
    ↪script-src 'self' https://$idp_host 'unsafe-eval'; img-src 'self' data:_
    ↪https://$idp_host; style-src 'self' 'unsafe-inline'; font-src 'self' data;:_
    ↪frame-src 'self'; connect-src 'self'";

    location ~ <path to login page of the application>$ {
        #if ($http_referer ~* "/blitz/simple") {
        #    set $idp_post_login "1";
        #}
        if ($http_referer ~* "<main server domain name>") {
            set $idp_post_login "1";
        }
        if ($idp_post_login = "1" ) {
            proxy_pass http://oc-web$request_uri;
        }
        if ($idp_post_login = "0" ) {
            proxy_pass http://$idp_host/blitz/simple/prepare$request_uri;
            break;
        }
    }
    location ~ /logout$ {
        if ($activLogout = "1") {
            return 302 https://<main server domain name>/blitz/simple/active_
            ↪logout?app=$host;
        }
        proxy_pass http://oc-web$request_uri;
    }
}

```

(continues on next page)

(continued from previous page)

```

}
location / {
    proxy_pass http://oc-web;
}
}

```

### 2.3.6 Interaction via the REST API

To invoke the REST services of Blitz Identity Provider, you must configure an application that will act as a client system for the REST services. To do this, [register a new application](#) (page 158) in Applications section.

Then go to the application settings, specify REST as the connection protocol and fill in the following data:

- `Password` - will be used during HTTP Basic authentication, as `login` - the client system identifier; if the parameter is not set, HTTP Basic authentication will not be possible for this client system;
- `Permissible CN` - list of values of CN attribute of the certificate used in TLS authentication; if no parameters are set, TLS authentication will not be possible for this client system.

The screenshot shows the 'Protocols' configuration page. At the top, there are tabs for 'SAML', 'OAuth 2.0', 'Simple', and 'REST'. The 'REST' tab is active. Below the tabs, there is a text instruction: 'To authenticate application for accessing Blitz Identity Provider API using Basic-authentication specify the password. As login using the application identifier'. There is a 'Password' input field with a masked password and a toggle for visibility. Below that, another instruction: 'To use TLS-authentication specify permissible CN from the application's certificate. You can specify several values'. There is a 'Permissible CN' input field with a placeholder text 'Specify Common Name (CN) and press Enter'. At the bottom, there is a blue 'Save' button.

If the application is not configured with a REST connection protocol, then it will not be able to use the REST APIs of Blitz Identity Provider server that are secured by HTTP Basic Authorization.

### 2.3.7 Access to network services via RADIUS

It is possible to configure the connection of users to network access points (RDP, VPN, Wi-Fi, etc.) using the RADIUS protocol. The connection setup is performed in the sequence described below.

#### RADIUS Help

Remote Authentication Dial In User Service (RADIUS) [RFC 2865<sup>44</sup>](https://datatracker.ietf.org/doc/html/rfc2865) is a protocol used for centralized management of authorization, authentication, and accounting for access to network services and equipment. This protocol is used to communicate between the server and the RADIUS client. After the user requests access to the network service, the corresponding client sends a request to the server, as a result of which the server checks the presence of the user in the database. If the user is found, the server sends the client permission to authenticate him.

The RADIUS server is Blitz Identity Provider, the client is a connected network service. In the current implementation, the server searches for users in all connected repositories. Network services are configured in Blitz Identity Provider as an application.

The server supports the following authentication methods:

<sup>44</sup> <https://datatracker.ietf.org/doc/html/rfc2865>

- the first factor: login and password;
- the second factor: confirmation by code from SMS, PUSH, TOTP, HOTP, email, or through the User Profile.

### Step 1. Configure the RADIUS Server

To configure the RADIUS server in Blitz Identity Provider, follow these steps:

1. In the admin console, go to RADIUS.
2. Configure the server configuration sequentially.

#### General settings

This tab specifies the general settings of the RADIUS server.

- `Status`: enabling the server.
- `Network binding address`: a list of addresses from which the server processes requests.

---

**Tip:** To process requests from all available network interfaces, set `0.0.0.0`.

---

- `Network port`: The RADIUS port to which requests are received. If the port is not specified, then port 1812 is used.
- `Maximum number of requests processed`: the maximum number of requests processed by the server at the same time (the rest are discarded).
- `Second factor timeout`: The time in seconds that is given to the user to pass the second factor.

**Attention:** This time must be agreed with the RADIUS client due to the correct setting of the waiting time for the RADIUS server response.

RADIUS server configuration

General settings   Network segments   Request processing flows

Server settings

Status

Network binding address   
Only requests from the specified address are processed. Enter `0.0.0.0` to process requests from all available network interfaces

Network port   
Requests are accepted on this port. If the port is not specified, then port 1812 is used

Maximum number of requests processed   
Maximum number of requests the server can process simultaneously (others are discarded)

Second factor timeout   
Determines how many seconds the user has to complete the second factor. This time must be matched with the RADIUS client by correctly setting its timeout

Click Save.

## Network segments

The identification of applications is carried out by network segments. Specify the subnet, the shared key, and the default application so that the request from this subnet is associated with this application. If several applications request authentication from the same subnet, they can be identified by the `NasId`.

**Attention:** Subnets with a narrower prefix have priority.

- **Name:** Enter a custom name for the network segment.
- **Subnet:** Enter the prefix of the subnet from which requests will be associated with the application.
- **Shared key:** generate and enter the key that you will need to [enter on the side of a network service](#) (page 190).
- **Default application:** Select the application that the request from this subnet will be associated with. If there are several applications, it will act as the default application.
- **Matching of `NasId` and applications:** if it is assumed that several applications will request authentication from the same subnet, set the `NasId`, by which the RADIUS server will identify them.

RADIUS server configuration

General settings **Network segments** Request processing flows

Network segment parameters

Name

Subnet

Shared key

Default Application

NasId and application mapping not configured

[add](#)

Click Save.

## Request processing procedures

This tab contains a list of Java procedures that will process requests from connected applications. The procedures determine the authentication factor and implement other network access policies. In the simplest case, the procedures include the first or second factor. You can create several procedures depending on the security requirements of different network points.

To create a request processing procedure, follow these steps:

1. Click Create a new request processing flow.
2. Specify the settings:
  - **Status:** enabling the procedure.
  - **Flow identifier:** Specify the procedure ID.

**Attention:** The Java class describing the request processing procedure should have the same name.

- **Description:** Enter a description of the procedure.

**RADIUS server configuration**

General settings   Network segments   **Request processing flows**

Request processing flow

Status

Flow identifier

The Java class that describes the request processing flow must have the same name

Description

Cancel   Create

3. Click Save.

4. Enter the source code of the procedure:

- Control the processing of RADIUS requests, you need to write a class in Java that implements the `RadiusFlow` interface.
- If the second authentication factor is used, call `RadiusResult.more("method")`, where `method` takes one of the following values: `sms`, `push`, `totp`, `hotp`, `email`, `prfc` (confirmation in the User Profile).

**Note:** When confirming through the User Profile, a message about the login attempt appears in it, in which the user must click Confirm.

**Attention:** In order for the factor to work, the User Profile must be opened with the mandatory passage of two authentication factors.

Listing 11: An example of the 2FA procedure via confirmation in your User Profile

```
package com.identityblitz.idp.radius.flow;

public class RadTest2 implements RadiusFlow {

    public String loginN12(final String login) {
        return login;
    }
}
```

(continues on next page)



(continued from previous page)

```

public RadiusResult next(final RadiusContext context) {
    if (context.factor() == 1) {
        //return RadiusResult.more("sms");

        return RadiusResult.more("prfc");
    }

    return RadiusResult.authenticated(context.subject());
}
}

```

- If the first factor is used, deactivate the `if (context.factor() == 1)`.

Listing 12: Example of the 1FA procedure

```

package com.identityblitz.idp.radius.flow;

public class TestRadius implements RadiusFlow {

    public String loginN12(final String login) {
        return login;
    }

    public RadiusResult next(final RadiusContext context) {

        return RadiusResult.authenticated(context.subject());
    }
}

```

- You can invoke the confirmation method selector by using `RadiusResult.challenge` in the procedure, as well as show an instruction on how to pass the second factor authentication by using `RadiusResult.dialog`.

```

private final Logger logger = LoggerFactory.getLogger("com.
↪identityblitz.idp.flow.radius");

public String loginN12(final String login) {
    return login;
}

public RadiusResult next(final RadiusContext context) {
    if (context.factor() == 1) {
        return RadiusResult.challenge(Challenges.password());
    }
    return RadiusResult.authenticated(context.subject());
}

public RadiusResult dialog(final RadiusContext context,
                           final String message,
                           final java.util.Map<String, String>↪
↪answers,
                           final String answer) {
    if(message.equals("challengeChoose")) {
        final String challenge = answers.get(answer);
        if(challenge != null) return RadiusResult.challenge(Challenges.
↪byName(challenge));
        else return RadiusResult.dialog(message, answers);
    }
}

```

(continues on next page)

(continued from previous page)

```

    } else {
      return RadiusResult.rejected("unsupportedMessage");
    }
  }
}

```

5. To compile, click Save.

## Step 2. Configure the application

To configure the application, follow these steps:

- In the admin console, go to Applications. [Create an](#) (page 157) application with basic settings.
  - Identifier (entityID or client\_id),
  - Name,
  - Domain: the domain of the network service.

### New application

Identifier (entityID  
or client\_id)

Application identifier. Used for identifying application within the SAML (corresponds to entityID) and OAuth 2.0 (corresponds to client\_id) protocols.

Name

Human-readable application name. Is used only inside Blitz Identity Provider.

Domain

Click Save.

- In the section Protocols of the application on the tab RADIUS set the following settings:
  - Check the box `The password is checked by the application itself if Blitz Identity Provider will be used for the second authentication factor.`
  - `Second factor timeout:` The time in seconds that is given to the user to pass the second factor. If the parameter is omitted, the value will be taken from the RADIUS server settings.

**Attention:** This time must be agreed with the RADIUS client due to the correct setting of the waiting time for the RADIUS server response.

- Select the procedure for processing requests from the application. In the list `Processing flow` displays all [procedures created by](#) (page 185) on the RADIUS server.

**Attention:** Carefully configure [integration](#) (page 190) on the network service side. If the `NasId` is not defined in the requests coming from the application, the application is recognized by Blitz Identity Provider as the default application for this network segment, even if they are actually different applications. In this case, the request processing procedure that is set for the default application will be performed, and not the one that is selected.

## Protocols

SAML

OAuth 2.0

Simple

REST

RADIUS

Block of specific application settings for the RADIUS protocol

The password is checked by the application itself

Second factor  
timeout

Determines how many seconds the user has to complete the second factor. This time must be matched with the RADIUS client by correctly setting its timeout. The default value is taken from the general RADIUS protocol settings

Processing flow

radius

If identifier is not specified, basic processing flow is used

Click Save.

### Step 3. Configuration on the network service side

To complete the connection, enter the following settings on the network service side:

- IP address of the server with `blitz-idp`.
- The shared key specified in the settings [network segment](#) (page 185) corresponding to the application (network service) on the RADIUS server. Using this key, the server will identify the network service and run the access processing procedure selected for it.
- `NasId` (if necessary).
- The waiting time for a response from the RADIUS server, corresponding to the waiting time set on the server for the second factor.




## 2.4 Customization with Java code

### 2.4.1 Login procedures and their creation

#### About the login procedures

Java authentication flows are used to configure the rules for user access to different applications. The authentication flows can be used to determine, for example, which applications should be available to which users, under what conditions two-factor authentication should be required, and which login validation methods a user can use. The use of authentication flows allows an organization to enforce its application access control policies.

Authentication flows are managed in the section Login procedures of the Blitz Identity Provider admin console.

Configured authentication flows			
Identifier	Applications	Description	Status
FFmethods v1		Limited list of first factor methods for application	Not activated 
Require2ndFactor v1		This procedures enables 2nd factor for the application	Not activated 
AccessByAttribute v1		If the user attribute "appList" (as an array) contains entityID (or client_id) of the application, access will be granted	Not activated 

[Create new authentication flow](#)

#### Creating a procedure

Creation of an authentication flow has following steps:

1. Specifying the basic parameters of the flow:
  - flow ID;
  - flow description;
  - applications - a list of applications that will use the authentication flow.

**Important:** Only one flow can be created for each application. If no flow is created for a given application, the standard entry procedure (default authentication flow) will be applied to that application. If a flow is created without specifying applications, it will replace the standard authentication flow.

[Create a new authentication flow](#)

Flow ID   
Flow ID should be a correct Java identifier. Java class that defines this flow will have the same name

Description   
If necessary enter a comment that describes the features and flow aim.

Applications   
List of applications that will use this authentication flows. If applications are not defined this flow will be regarded as global and will be applied to all cases when application is not specified. You can activate only one global procedure, and there shouldn't be any collisions when defining the authentication flow for any application.

[Create](#)

2. Writing the source code of the procedure. For successful operation of the authentication flow it is necessary to write a Java class that implements the necessary `Strategy` interface. All context information about the user, the current state of the authentication flow, etc. is available in the `Context` object. The flow consists of two blocks that define:
  - actions taken at the initial stage of the authentication process. In this block, for example, it is possible to define under what conditions to switch to the application in SSO mode (if the user has been previously authenticated);
  - actions taken after the initial authentication of the user. In this block, for example, you can define which two-factor authentication methods to use under which conditions.
3. After writing the code, you should press the “*Compile*” button. If errors are detected, incorrect code fragments will be highlighted and signed with errors.
4. If the compilation was successful you can save the flow.
5. The saved procedure can be activated by clicking on the “*Activate*” button in the header of the corresponding procedure.
6. Both activated and deactivated procedures can be edited. After editing, compile the procedure and then save it. If it has been activated, the new compiled flow will replace the old one.

**Warning:** If the procedure has been activated, only the compiled one can be saved. In other words, if an error while editing an activated flow has been detected, the “*Save*” button will not work and after reloading the page all changes will be lost.

## Procedure source code

To make an authentication flow you should program a [Java](#) class, implementing interface [Strategy](#). Class name must match the flow ID ([FFmethods](#)). Class must have a public [default](#) constructor. For security reasons the class is loaded by a separate [class loader](#) with limited list of [imports](#). All contextual information about the user, the current status of the authentication flow, etc. is available in the object [Context](#).

View interface Strategy ▾

View permitted imports ▾

View Context description ▾

Load Blitz Development Kit

```

1 package com.identityblitz.idp.flow.dynamic;
2
3 import java.lang.*;
4 import java.util.*;
5 import org.slf4j.LoggerFactory;
6 import org.slf4j.Logger;
7 import com.identityblitz.idp.login.authn.flow.Context;
8 import com.identityblitz.idp.login.authn.flow.Strategy;
9 import com.identityblitz.idp.login.authn.flow.StrategyState;
10 import com.identityblitz.idp.flow.dynamic.*;
11
12 import static com.identityblitz.idp.login.authn.flow.StrategyState.*;
13
14 public class FFmethods implements Strategy {
15     private final Logger logger = LoggerFactory.getLogger("com.identityblitz.idp.flow.dynamic");
16
17
18     @Override public StrategyState begin(final Context ctx) {
19         if(ctx.claims("subjectId") != null)
20             return StrategyState.ENOUGH;
21         else
22             return StrategyState.MORE(new String[]{"password","x509"});
23     }
24 }
25
26 @Override public StrategyState next(final Context ctx) {
27     String reqFactor = ctx.userProps("requiredFactor");
28     if(reqFactor == null)
29         return StrategyState.ENOUGH;
30     else {
31         if(Integer.valueOf(reqFactor) == ctx.justCompletedFactor())
32             return StrategyState.ENOUGH;
33         else
34             return StrategyState.MORE(new String[]{});
35     }
36 }
37 }

```

Compile

Save

## 2.4.2 Ready-made login procedures

The package includes several ready-made procedures that can be changed if necessary:

- [forced two-factor application authentication in](#) (page 194) application ([Require2ndFactor](#));
- [limiting the list of available first factor methods when logging into the](#) (page 194) application ([FFmethods](#));
- [granting access to the application only with a certain value of the attribute](#) (page 195) ([AccessByAttribute](#));
- [prohibit logging into the application after the account expires](#) (page 197) ([AccountExpiresCheck](#));
- [allow logging into the application only from certain networks](#) (page 197) ([AllowedIPs](#));
- [prohibit work in several simultaneous sessions](#) (page 199) ([RestrictSessions](#));
- [saving a list of user groups in statements \(claims\)](#) (page 199) ([AddGroupsToToken](#));
- [displaying an announcement to the user at login](#) (page 200) ([InfoPipe](#));
- [request for user to enter attribute or update a phone number and email](#) (page 202) ([PipeAttrActAdd](#));
- [request for the user to enter a security question unless it is asked in the account](#) (page 205) ([PipeSecQuestion](#));
- [registration of security key WebAuthn, Passkey, FIDO2 at login](#) (page 206) ([PipeWebAuthn](#)).
- [display a list of value selections to the user at login](#) (page 208) ([ChoicePipe](#)).

Listings of these procedures are provided below. For ease of debugging, you can output information on the authentication state to the log using the `logger.debug()` function. For example, the following command will log the specified authentication level for a user:

```
logger.debug("requiredFactor="+ctx.userProps("requiredFactor"));
```

### Forced two-factor authentication

The `Require2ndFactor` procedure requires two-factor authentication to access the application. If a user goes to the application within a single session, if there is one factor passed, the user will have the second factor additionally verified, i.e., SSO will not work in this case.

```
public class Require2ndFactor implements Strategy {

    private final Logger logger = LoggerFactory.getLogger("com.identityblitz.idp.
↪flow.dynamic");

    @Override public StrategyBeginState begin(final Context ctx) {
        if (ctx.claims("subjectId") != null) {
            if (ctx.sessionTrack().split(",").length < 2)
                return StrategyState.MORE(new String[]{});
            else
                return StrategyState.ENOUGH();
        }
        else {
            return StrategyState.MORE(new String[]{});
        }
    }

    @Override public StrategyState next(final Context ctx) {
        if (ctx.justCompletedFactor() == 1)
            return StrategyState.MORE(new String[]{});
        else
            return StrategyState.ENOUGH();
    }
}
```

### Limiting the list of available first factor methods

The `FFmethods` procedure allows to offer only certain identification and authentication methods to the user when entering the application (a similar procedure with a different list of methods can be assigned to another application). The procedure uses the following identifiers to designate the first factor authentication methods:

- `password` - login using login and password;
- `x509` - login via electronic signature;
- `externalIdps` - login via external identity providers (social networks etc.);
- `spnego` - login via operating system session;
- `sms` - login via confirmation code from SMS.
- `knownDevice` - login via known device;
- `qrCode` - login via QR code;
- `webAuthn` - login with security keys (WebAuthn, Passkey, FIDO2);
- `tls` – login based on the transmitted HTTP header.

```

public class FFmethods implements Strategy {

    private final Logger logger = LoggerFactory.getLogger("com.identityblitz.idp.
↳flow.dynamic");

    @Override public StrategyBeginState begin(final Context ctx) {
        if(ctx.claims("subjectId") != null)
            return StrategyState.ENOUGH();
        else
            return StrategyState.MORE(new String[]{"password","x509"});
    }

    @Override public StrategyState next(final Context ctx) {
        Integer reqFactor = (ctx.user() == null) ? null : ctx.user().
↳requiredFactor();
        if(reqFactor == null || reqFactor == 0)
            return StrategyState.ENOUGH();
        else {
            if(reqFactor == ctx.justCompletedFactor())
                return StrategyState.ENOUGH();
            else
                return StrategyState.MORE(new String[]{});
        }
    }
}

```

### Log in only with a certain attribute value

The `AccessByAttribute` procedure uses the `appList` attribute to decide whether a user can access the application. This procedure requires the `appList` attribute to be created as an array (Array of strings). Application identifiers should be used as the values of the elements of this array. As a result, access to an application will be granted if among the values of `appList` a given user has the identifier of this application. This procedure architecture allows you to assign it to several applications at once and control access to them using a single attribute.

```

public class AccessByAttribute implements Strategy {

    private final Logger logger = LoggerFactory.getLogger("com.identityblitz.idp.
↳flow.dynamic");

    @Override public StrategyBeginState begin(final Context ctx) {
        if(ctx.claims("subjectId") != null){
            int appListIdx = 0;
            boolean hasAccess = false;
            while (appListIdx > -1) {
                String app = ctx.claims("appList.[ " + appListIdx + " ]");
                logger.debug("app [ " + appListIdx + " ] = " + app);
                if (app == null){ appListIdx = -1; }
                else if (app.equals(ctx.appId())) { appListIdx = -1; hasAccess =
↳true; }
                else { appListIdx ++; logger.debug("AppList index = " +
↳appListIdx); }
            }
            if(hasAccess)
                return StrategyState.ENOUGH();
            else
                return StrategyState.DENY;
        }
        else
    }
}

```

(continues on next page)



(continued from previous page)

```

        return StrategyState.MORE(new String[]{});
    }

    @Override public StrategyState next(final Context ctx) {
        int appListIdx = 0;
        boolean hasAccess = false;
        while (appListIdx > -1) {
            String app = ctx.claims("appList." + appListIdx + "");
            logger.debug("app [" + appListIdx + "] = " + app);
            if (app == null) { appListIdx = -1; }
            else if (app.equals(ctx.appId())) { appListIdx = -1; hasAccess = true;
↪ }
                else { appListIdx++; logger.debug("AppList index = " + appListIdx); }
        }
        if (!hasAccess)
            return StrategyState.DENY;
        Integer reqFactor = 0;
        if (ctx.user() != null) {
            reqFactor = ctx.user().requiredFactor();
        }
        if (reqFactor == 0)
            return StrategyState.ENOUGH();
        else {
            if (reqFactor == ctx.justCompletedFactor())
                return StrategyState.ENOUGH();
            else
                return StrategyState.MORE(new String[]{});
        }
    }
}

```

An example of a simplified version of the procedure is to grant a user access to an application provided his e-mail address is `ivanov@company.ru`:

```

@Override public StrategyBeginState begin(final Context ctx) {
    if (ctx.claims("subjectId") != null) {
        if ("ivanov@company.ru".equals(ctx.claims("email")))
            return StrategyState.ENOUGH();
        else
            return StrategyState.DENY;
    }
    else
        return StrategyState.MORE(new String[]{});
}

@Override public StrategyState next(final Context ctx) {
    if (!"ivanov@company.ru".equals(ctx.claims("email")))
        return StrategyState.DENY;
    Integer reqFactor = (ctx.user() == null) ? null : ctx.user().requiredFactor();
    if (reqFactor == null)
        return StrategyState.ENOUGH();
    else {
        if (reqFactor == ctx.justCompletedFactor())
            return StrategyState.ENOUGH();
        else
            return StrategyState.MORE(new String[]{});
    }
}

```

### Prohibiting login after account expiration

The `AccountExpiresCheck` procedure uses the `accountExpires` attribute to decide whether a user has access to the application. For this procedure to work, you must create an attribute `accountExpires` with the type `string` (`String`). In this attribute it is necessary to store the date (in the format `yyyy-MM-dd HH:mm`, for example `2021-09-23 13:58`), after which the access to the application will be blocked for this user. If the attribute value is not specified, the user will be allowed to enter the application.

```
public class AccountExpiresCheck implements Strategy {

    private final Logger logger = LoggerFactory.getLogger("com.identityblitz.idp.
↪flow.dynamic");

    @Override public StrategyBeginState begin(final Context ctx) {
        if ("login".equals(ctx.prompt())) {
            List<String> methods = new ArrayList<String>(Arrays.asList(ctx.
↪availableMethods()));
            methods.remove("cls");
            return StrategyState.MORE(methods.toArray(new String[0]), true);
        } else {
            if(ctx.claims("subjectId") != null)
                return StrategyState.ENOUGH();
            else
                return StrategyState.MORE(new String[]{});
        }
    }

    @Override public StrategyState next(final Context ctx) {
        if (ctx.claims("accountExpires") != null && isExpired(ctx.claims("accountExpires
↪"))))
            return StrategyState.DENY("account_expired", true);
        Integer reqFactor = (ctx.user() == null) ? null : ctx.user().requiredFactor();
        if(reqFactor == null || reqFactor == ctx.justCompletedFactor())
            return StrategyState.ENOUGH();
        else
            return StrategyState.MORE(new String[]{});
    }

    public static boolean isExpired(String strData) {
        try {
            Date now = new Date();
            Date date = new SimpleDateFormat("yyyy-M-d HH:mm").parse(strData);
            return now.after(date);
        } catch (ParseException e) {
            throw new RuntimeException(e);
        }
    }
}
```

### Log in only from certain networks

The `AllowedIPs` procedure uses the `ALLOW_IP` constant to decide whether the user can access the application. In this constant it is necessary to specify the list of networks from which the access to the application is possible, it is acceptable to specify several networks. When entering the application, the user's IP address will be checked to see if it matches one of the values included in the constant. If it matches, the user will be allowed to enter the application, if it does not match - access will be denied.

```
public class AllowedIPs implements Strategy {
```

(continues on next page)

(continued from previous page)

```

private final Logger logger = LoggerFactory.getLogger("com.identityblitz.idp.
↪flow.dynamic");
private final static String[] ALLOW_IP = {"179.218","180.219"};

@Override public StrategyBeginState begin(final Context ctx) {
    if ("login".equals(ctx.prompt())){
        List<String> methods = new ArrayList<String>(Arrays.asList(ctx.
↪availableMethods()));
        methods.remove("cls");
        return StrategyState.MORE(methods.toArray(new String[0]), true);
    } else {
        if(ctx.claims("subjectId") != null)
            return StrategyState.ENOUGH();
        else
            return StrategyState.MORE(new String[]{});
    }
}

@Override public StrategyState next(final Context ctx) {
    if (!_allowed_ip(ctx.ip())) {
        return StrategyState.DENY("ip_not_allowed", true);
    }
    Integer reqFactor = (ctx.user() == null) ? null : ctx.user().
↪requiredFactor();
    if(reqFactor == null || reqFactor == ctx.justCompletedFactor()) {
        return StrategyState.ENOUGH_BUILDER()
            .build();
    } else
        return StrategyState.MORE(new String[]{});
}

private Boolean _allowed_ip(final String IP) {
    int IpListIdx = 0;
    boolean ipAllowed = false;
    while (IpListIdx > -1) {
        String ip_part = ALLOW_IP[IpListIdx];
        if (IP.startsWith(ip_part)) {
            ipAllowed = true;
            IpListIdx = -1;
        } else if (ALLOW_IP.length == (IpListIdx + 1)) {
            IpListIdx = -1;
        } else {
            IpListIdx ++;
        }
    }
    return ipAllowed;
}
}

```

## Prohibition of work in several simultaneous sessions

The `RestrictSessions` procedure prohibits working in multiple sessions.

```
public class RestrictSessions implements Strategy {

    private final Logger logger = LoggerFactory.getLogger("com.identityblitz.idp.
↳flow.dynamic");

    @Override public StrategyBeginState begin(final Context ctx) {
        List<String> methods = new ArrayList<String>(Arrays.asList(ctx.
↳availableMethods()));
        if ("login".equals(ctx.prompt())){
            methods.remove("cls");
            return StrategyState.MORE(methods.toArray(new String[0]), true);
        } else {
            if(ctx.claims("subjectId") != null)
                return StrategyState.ENOUGH();
            else {
                methods.remove("cls");
                return StrategyState.MORE(methods.toArray(new String[0]));
            }
        }
    }

    @Override public StrategyState next(final Context ctx) {
        Integer reqFactor = (ctx.user() == null) ? null : ctx.user().
↳requiredFactor();
        if(reqFactor == null || reqFactor == ctx.justCompletedFactor()) {
            return StrategyState.ENOUGH_BUILDER().singleSession(true).build();
        } else
            return StrategyState.MORE(new String[]{});
    }
}
```

## Saving a list of user groups in claims

The `AddGroupsToToken` procedure records a list of user groups in the `grps` statement. For this procedure to work, the conditions must be met:

- `memberOf` attribute is configured to display the user's groups;
- session statement `grps` (see [Adding attributes to an identity token](#) (page 177)) was added to the configuration file.

When logging into the application, it will check if the user has groups in the `memberOf` attribute, and if they are present there, they will be added to the `grps` statement.

```
public class AddGroupsToToken implements Strategy {

    private final Logger logger = LoggerFactory.getLogger("com.identityblitz.idp.
↳flow.dynamic");

    @Override public StrategyBeginState begin(final Context ctx) {
        if ("login".equals(ctx.prompt())){
            List<String> methods = new ArrayList<String>(Arrays.asList(ctx.
↳availableMethods()));
            methods.remove("cls");
            return StrategyState.MORE(methods.toArray(new String[0]), true);
        } else {
            if(ctx.claims("subjectId") != null)
```

(continues on next page)

(continued from previous page)

```

        return StrategyState.ENOUGH();
    else
        return StrategyState.MORE(new String[]{});
    }
}

@Override public StrategyState next(final Context ctx) {
    Integer reqFactor = (ctx.user() == null) ? null : ctx.user().
↪requiredFactor();
    if(reqFactor == null || reqFactor == ctx.justCompletedFactor()) {
        List<String> grps = new ArrayList<String>();
        int groupListIdx = 0;
        while (groupListIdx > -1) {
            String group = ctx.claims("memberOf.[" + groupListIdx + "]");
            logger.debug("### group [" + groupListIdx + "] = " + group);
            if (group == null) {
                groupListIdx = -1;
            } else {
                grps.add(ctx.claims("memberOf.[" + groupListIdx + "]"));
                groupListIdx ++;
            }
        }
        LClaimsBuilder claimsBuilder = ctx.claimsBuilder();
        if (grps.size() > 0) {
            claimsBuilder.addClaim("grps", grps);
        }
        LClaims claims = claimsBuilder.build();
        return StrategyState.ENOUGH_BUILDER()
            .withClaims(claims)
            .build();
    } else
        return StrategyState.MORE(new String[]{});
}
}

```

### Displaying an announcement to the user at login

You can configure Blitz Identity Provider to show an announcement to the user upon login. This can show the user one or two buttons, and the user's choice can be analyzed in the login procedure.

#### Procedure

The `InfoPipe` procedure allows ads to be shown to the user at 30-day intervals when they log in. The following changes must be made to the procedure before it can be used:

- in the `requiredNews()` function, adjust the criteria for displaying the ad - for example, in the example it is set to show once every 30 days if the user clicked the refuse button last time the ad was displayed;
- in the `DOMAIN` constant, specify the URI at which Blitz Identity Provider is accessible from the user's browser;
- [настройка](#) (page 202) notification type in the configuration file;
- [configure](#) (page 241) notification text and button names in messages.

```

public class InfoPipe implements Strategy {

    private final Logger logger = LoggerFactory.getLogger("com.identityblitz.idp.

```

(continues on next page)

(continued from previous page)

```

↔flow.dynamic");
    private final static String DOMAIN = "example.com";

    @Override public StrategyBeginState begin(final Context ctx) {
        if ("login".equals(ctx.prompt())){
            List<String> methods = new ArrayList<String>(Arrays.asList(ctx.
↔availableMethods()));
            methods.remove("cls");
            return StrategyState.MORE(methods.toArray(new String[0]), true);
        } else {
            if(ctx.claims("subjectId") != null)
                return StrategyState.ENOUGH();
            else
                return StrategyState.MORE(new String[]{});
        }
    }

    @Override public StrategyState next(Context ctx) {
        if (ctx.user() == null || ctx.user().requiredFactor() == null ||
            ctx.user().requiredFactor().equals(ctx.justCompletedFactor()))
            if (requiredNews("user_agreement", ctx)) return showNews("user_
↔agreement", ctx);
            else return StrategyState.ENOUGH();
        else
            return StrategyState.MORE(new String[] {});
    }

    private boolean requiredNews(final String pipeId, final Context ctx) {
        Long readOn = ctx.user().userProps().numProp("pipes.info." + pipeId + ".
↔disagreedOn");
        return (readOn == null || Instant.now().getEpochSecond() - readOn >=
↔30*86400);
    }

    private StrategyState showNews(final String pipeId, final Context ctx) {
        String uri = "https://" + DOMAIN + "/blitz/pipes/info/start?&pipeId=" +
↔pipeId + "&appId=blitz_profile";
        Set<String> claims = new HashSet<String>(){
            add("instanceId");
        };
        Set<String> scopes = new HashSet<String>(){
            add("openid");
        };
        return StrategyState.ENOUGH_BUILDER()
            .withPipe(uri, "<CLIENT_ID>", scopes, claims)
            .build();
    }
}

```

### Adding a procedure to blitz.conf

in the `blitz.conf` configuration file add the `blitz.prod.local.idp.built-in-pipes` section, in which assign the `id` identifier specified in the procedure and the `type` announcement type to the auxiliary application with the `info` type. The following announcement configurations are possible:

- `news` - a single button is displayed,
- `agreement` - two buttons are displayed.

Example configuration of two `info` helper applications with identifiers `alarm` and `user_agreement`:

```
"built-in-pipes": {
  "info": [
    {
      "id": "alarm",
      "type": "news"
    },
    {
      "id": "user_agreement",
      "type": "agreement"
    }
  ]
}
```

### Request for user to enter attribute or actualize phone and email

The `PipeAttrActAdd` procedure allows to request the user to enter the attribute value. For cell phone and for email, periodic updating of the contact is implemented. For regular attribute (in the example `family_name` is used) one-time filling of the attribute. In case the user did not want to fill the attribute, the next request to enter the attribute after a certain time will be realized.

The following modifications must be made to the procedure before use:

- in the `DOMAIN` constant, specify the URI at which Blitz Identity Provider is accessible from the user's browser;
- in the constants `MOBILE_ATTR`, `EMAIL_ATTR`, `COMMON_ATTR` specify the names of the attributes to be filled in;
- in the `SKIP_TIME_IN_SEC` constant specify the time, not more often than which the user will be offered to fill the attribute;
- in the `ACT_TIME_IN_SEC` constant specify the time, not more often than which the user will be offered to update phone or email;
- in the `ASK_AT_1ST_LOGIN` constant, change the value if the request to fill in the contact should be performed at the first login (usually the first login occurs immediately after the account registration, so the setting is made so that the user is not prompted to fill in the data at the first login);
- in the body of the procedure instead of `_blitz_profile` specify the identifier of another application, if the attributes change should be made from an application other than the user profile;
- set texts in messages for attribute from `COMMON_ATTR` (default texts for email and phone can also be adjusted) - see `:ref:config-pipes-messages`. [Auxiliary application messages \(pipes\)](#) (page 241).

```
public class PipeAttrActAdd implements Strategy {

    private final Logger logger = LoggerFactory.getLogger("com.identityblitz.idp.
↪flow.dynamic");
    private final static String DOMAIN = "example.com";
    private final static String MOBILE_ATTR = "phone_number";
```

(continues on next page)

(continued from previous page)

```

private final static String EMAIL_ATTR = "email";
private final static String COMMON_ATTR = "family_name";
private final static Integer SKIP_TIME_IN_SEC = 30*86400;
private final static Integer ACT_TIME_IN_SEC = 30*86400;
private final static Boolean ASK_AT_1ST_LOGIN = false;

@Override public StrategyBeginState begin(final Context ctx) {
    if ("login".equals(ctx.prompt())){
        List<String> methods = new ArrayList<String>(Arrays.asList(ctx.
↪availableMethods()));
        methods.remove("cls");
        return StrategyState.MORE(methods.toArray(new String[0]), true);
    } else {
        if(ctx.claims("subjectId") != null)
            return StrategyState.ENOUGH();
        else
            return StrategyState.MORE(new String[]{});
    }
}

@Override public StrategyState next(final Context ctx) {
    Instant instant = Instant.now();
    Boolean new_device = false;
    if (ctx.ua().getNewlyCreated() && ctx.justCompletedFactor() == 1 && !ASK_
↪AT_1ST_LOGIN){
        logger.debug("User with sub={} is signing in, pid={}, on a new device",
            ctx.claims("subjectId"), ctx.id());
        new_device = true;
    }
    Integer reqFactor = ctx.user().requiredFactor();
    if(reqFactor == null || reqFactor == ctx.justCompletedFactor()) {
        Enough.Builder en_builder = StrategyState.ENOUGH_BUILDER();
        if (MOBILE_ATTR !=null && !new_device && requireActualizeAttr(MOBILE_
↪ATTR, ctx)) {
            String uri = "https://" +DOMAIN+"/blitz/pipes/attr/act?attr="
                +MOBILE_ATTR+"&canSkip=true&appId=_blitz_profile&verified=true
↪";
            Set<String> clms = new HashSet<String>(){
                add("instanceId");
                add(MOBILE_ATTR);
            };
            Set<String> scps = new HashSet<String>(){
                add("openid");
            };
            logger.debug("User has no {} or a non-actualized {}, so opening pipe
↪",
                MOBILE_ATTR, MOBILE_ATTR);
            en_builder = en_builder.withPipe(uri, "_blitz_profile", scps,
↪clms);
        } else if (EMAIL_ATTR !=null && !new_device &&
↪requireActualizeAttr(EMAIL_ATTR, ctx)) {
            String uri = "https://" +DOMAIN+"/blitz/pipes/attr/act?attr="
                +EMAIL_ATTR+"&canSkip=true&appId=_blitz_profile&verified=true";
            Set<String> clms = new HashSet<String>(){
                add("instanceId");
                add(EMAIL_ATTR);
            };
            Set<String> scps = new HashSet<String>(){
                add("openid");
            };
            logger.debug("User has no {} or a non-actualized {}, so opening pipe

```

(continues on next page)



(continued from previous page)

```

↪",
        EMAIL_ATTR, EMAIL_ATTR);
        en_builder = en_builder.withPipe(uri, "_blitz_profile", scps, ↪
↪clms);
        } else if (COMMON_ATTR !=null && !new_device &&
            requireActualizeAttr(COMMON_ATTR, ctx)) {
            String uri = "https://" + DOMAIN + "/blitz/pipes/attr/act?attr="
                + COMMON_ATTR + "&canSkip=true&appId=_blitz_profile";
            Set<String> clms = new HashSet<String>(){{
                add("instanceId");
                add(COMMON_ATTR);
            }};
            Set<String> scps = new HashSet<String>(){{
                add("openid");
            }};
            logger.debug("User has no {}, so opening pipe", COMMON_ATTR);
            en_builder = en_builder.withPipe(uri, "_blitz_profile", scps, ↪
↪clms);
        }
        return en_builder.build();
    } else {
        return StrategyState.MORE(new String[]{});
    }
}

private Boolean requireActualizeAttr(final String attrName, final Context ctx)
↪{
    if (attrName.equals(MOBILE_ATTR) && (ctx.passedTrack().startsWith("1:sms") ↪
↪||
        ctx.passedTrack().endsWith("sms"))) {
        logger.debug("User subjectId = {}, pid = {} used SMS, so no ↪
↪actualization needed",
            ctx.claims("subjectId"), ctx.id());
        return false;
    }
    if (attrName.equals(EMAIL_ATTR) && ctx.passedTrack().endsWith("email")) {
        logger.debug(
            "User subjectId = {}, pid = {} used EMAIL while auth, so no ↪
↪actualization needed",
            ctx.claims("subjectId"), ctx.id());
        return false;
    }
    Long skpTime = null;
    Long actTime = null;
    long now = Instant.now().getEpochSecond();
    if (ctx.user().userProps().numProp("pipes.act."+attrName+".skippedOn") != ↪
↪null) {
        skpTime = ctx.user().userProps().numProp("pipes.act."+attrName+" ↪
↪skippedOn");
    }
    if (skpTime != null && ((now - skpTime) < SKIP_TIME_IN_SEC)) {
        logger.debug(
            "User subjectId = {}, pid = {} has skipped update '{}' only '{} ↪
↪seconds ago, no actualization needed", ctx.claims("subjectId"), ctx.id(), ↪
↪attrName, (now - skpTime));
        return false;
    }
    if (ctx.claims(attrName) == null) return true;
    else {
        if (ctx.user().attrsCfmTimes() != null) {
            actTime = ctx.user().attrsCfmTimes().get(attrName);

```

(continues on next page)

(continued from previous page)

```

    }
    if (actTime == null) return true;
    else {
        logger.debug(
            "User subjectId = {}, pid = {} has updated '{}' '{}' seconds_
↪ago, actualization needed = {}", ctx.claims("subjectId"), ctx.id(), attrName,
↪(now - actTime), ((now - actTime) > ACT_TIME_IN_SEC));
        return ((now - actTime) > ACT_TIME_IN_SEC);
    }
}
}
}
}

```

### Requesting the user to enter a security question

The “PipeSecQuestion” procedure checks whether the user has a security question. If the question is not asked, the procedure prompts the user to enter it.

The following modifications must be made to the procedure before use:

- in the `DOMAIN` constant, specify the URI at which Blitz Identity Provider is accessible from the user’s browser;
- in the “`CAN_SKIP`” constant, specify the display mode: `true` – the user can skip filling; `false` – the user must set the value of the security question to complete authentication.

```

public class PipeSecQuestion implements Strategy {

    private final Logger logger = LoggerFactory.getLogger("com.identityblitz.idp.
↪flow.dynamic");
    private final static String DOMAIN = "example.com";
    private final static Boolean CAN_SKIP = true;

    @Override public StrategyBeginState begin(final Context ctx) {
        if ("login".equals(ctx.prompt())){
            List<String> methods = new ArrayList<String>(Arrays.asList(ctx.
↪availableMethods()));
            methods.remove("cls");
            return StrategyState.MORE(methods.toArray(new String[0]), true);
        } else {
            if(ctx.claims("subjectId") != null)
                return StrategyState.ENOUGH();
            else
                return StrategyState.MORE(new String[]{});
        }
    }

    @Override public StrategyState next(final Context ctx) {
        Integer reqFactor = (ctx.user() == null) ? null : ctx.user().
↪requiredFactor();
        if (reqFactor == null || reqFactor.equals(ctx.justCompletedFactor())){
            if(requireAddSecQsn(ctx)) return addSecQsn(ctx);
            else return StrategyState.ENOUGH();
        }
        else return StrategyState.MORE(new String[]{});
    }

    private Boolean requireAddSecQsn(final Context ctx) {
        String secQsn = (ctx.user() == null) ? null : ctx.user().
↪securityQuestion();
    }
}

```

(continues on next page)

(continued from previous page)

```

        Long agreedOn = (ctx.user() == null) ? null : ctx.user().userProps().
↪numProp("pipes.addSecQsn.agreedOn");
        Long disagreedOn = (ctx.user() == null) ? null : ctx.user().userProps().
↪numProp("pipes.addSecQsn.disagreedOn");
        if (secQsn != null) return false;
        else if (disagreedOn == null) return true;
        else {
            long now = Instant.now().getEpochSecond();
            return ((now - disagreedOn) > 1);
        }
    }

    private StrategyState addSecQsn(final Context ctx) {
        String uri = "https://" + DOMAIN + "/blitz/pipes/secQsn/start?canSkip="+CAN_
↪SKIP+"&appId=_blitz_profile";
        Set<String> claims = new HashSet<String>(){{
            add("instanceId");
        }};
        Set<String> scopes = new HashSet<String>(){{
            add("openid");
        }};
        return StrategyState.ENOUGH_BUILDER()
            .withPipe(uri, "_blitz_profile", scopes, claims)
            .build();
    }
}

```

### Registration of security key (WebAuthn, Passkey, FIDO2) at login

The PipeWebAuthn procedure allows you to request the user to register a security key (WebAuthn, Passkey, FIDO2) at login.

The following modifications must be made to the procedure before use:

- in the DOMAIN constant, specify the URI at which Blitz Identity Provider is accessible from the user's browser;
- in the SKIP\_TIME\_IN\_SEC constant specify the time, not more often than which the user will be offered to fill the attribute;
- in the ASK\_AT\_1ST\_LOGIN constant, change the value if the request for security key issuance should be performed at the first login (usually the first login occurs immediately after account registration, so the setting is made so that the user is not prompted to fill in the data at the first login);
- in the body of the procedure instead of `_blitz_profile` specify the identifier of another application, if the attributes change should be made from an application other than the user profile.

```

public class PipeWebAuthn implements Strategy {

    private final Logger logger = LoggerFactory.getLogger("com.identityblitz.idp.
↪flow.dynamic");
    private final static String DOMAIN = "example.com";
    private final static Integer SKIP_TIME_IN_SEC = 30*86400;
    private final static Boolean ASK_AT_1ST_LOGIN = true;

    @Override public StrategyBeginState begin(final Context ctx) {
        if ("login".equals(ctx.prompt())){
            List<String> methods = new ArrayList<String>(Arrays.asList(ctx.
↪availableMethods()));
            methods.remove("cls");
        }
    }
}

```

(continues on next page)

(continued from previous page)

```

        return StrategyState.MORE(methods.toArray(new String[0]), true);
    } else {
        if (ctx.claims("subjectId") != null)
            return StrategyState.ENOUGH();
        else
            return StrategyState.MORE(new String[]{});
    }
}

@Override
public StrategyState next(Context ctx) {
    Boolean new_device = false;
    if (ctx.ua().getNewlyCreated() && ctx.justCompletedFactor() == 1 && !ASK_
↪AT_1ST_LOGIN){
        logger.debug("User with sub={} is signing in, pid={}, on a new device",
            ctx.claims("subjectId"), ctx.id());
        new_device = true;
    }
    if (ctx.user() == null || ctx.user().requiredFactor() == null ||
        ctx.user().requiredFactor().equals(ctx.justCompletedFactor()))
        if (!new_device && requiredWebAuthn(ctx))
            return webAuthn(ctx);
        else
            return StrategyState.ENOUGH();
    else
        return StrategyState.MORE(new String[] {});
}

private boolean requiredWebAuthn(final Context ctx) {
    IBrowser br = ctx.ua().asBrowser();
    String deviceType = br.getDeviceType();
    String os = br.getOsName();
    List<WakMeta> keyList = null;
    logger.trace("User subjectId = {}, pid = {} is logging using device '{}' ↪
↪and OS '{}', checking configured webAuthn keys", ctx.claims("subjectId"), ctx.
↪id(), deviceType, os);
    ListResult<WakMeta> keys = ctx.dataSources().webAuthn().
↪keysOfCurrentSubject();
    if (keys != null) {
        keyList = keys.filter(k -> deviceType.equals(k.addedOnUA().
↪deviceType()))
            .filter(k -> os.equals(k.addedOnUA().osName())).list();
    }
    if (keys != null && keyList.size() > 0) {
        logger.debug("User subjectId = {}, pid = {} has '{}' webAuthn keys for ↪
↪device '{}' and OS '{}'", ctx.claims("subjectId"), ctx.id(), keyList.size(), ↪
↪deviceType, os);
        return false;
    } else {
        logger.debug("User subjectId = {}, pid = {} has no configured webAuthn ↪
↪keys for device '{}' and OS '{}'", ctx.claims("subjectId"), ctx.id(), deviceType,
↪os);
    }
    Long disagreedOn = ctx.user().userProps().numProp("pipes.addKey." + ↪
↪deviceType + "." + os + ".disagreedOn");
    if (disagreedOn == null) {
        return true;
    } else if (Instant.now().getEpochSecond() - disagreedOn > SKIP_TIME_IN_
↪SEC) {
        logger.debug("User subjectId = {}, pid = {} has skipped Webauthn '{}' ↪
↪seconds ago, so open webAuthn pipe", ctx.claims("subjectId"), ctx.id(), (Instant.

```

(continues on next page)

(continued from previous page)

```

↪now().getEpochSecond() - disagreedOn));
        return true;
    } else {
        logger.debug("User subjectId = {}, pid = {} has skipped Webauthn '{}'_
↪seconds ago, no need to open webAuthn pipe", ctx.claims("subjectId"), ctx.id(),
↪(Instant.now().getEpochSecond() - disagreedOn));
        return false;
    }
}

private StrategyState webAuthn(final Context ctx) {
    String uri = "https://" + DOMAIN + "/blitz/pipes/conf/webAuthn/start?&
↪canSkip=true&appId=_blitz_profile";
    Set<String> claims = new HashSet<String>() {{
        add("instanceId");
    }};
    Set<String> scopes = new HashSet<String>() {{
        add("openid");
    }};
    Map<String, Object> urParams = new HashMap<String, Object>();
    return StrategyState.ENOUGH_BUILDER()
        .withPipe(uri, "_blitz_profile", scopes, claims).build();
}
}

```

### Display a list of value selections to the user at login

You can configure after login Blitz Identity Provider to show the user a selection box from a list of values and store the result of the selection in an attribute in the user's account.

#### Procedure

The ChoicePipe procedure allows the user to show the value list selection pages on login. The following changes must be made to the procedure before it can be used:

- in the DOMAIN constant instead of <BLITZ-HOST> specify the URI where Blitz Identity Provider is accessible from the user's browser, and in the CLIENT\_ID constant instead of <CLIENT\_ID> specify the application identifier (with permissions to scope openid) on behalf of which the helper application will be executed;
- [set the notification type in the configuration file](#) (page 209);
- [set](#) (page 241) notification text and button names in messages.

```

public class ChoicePipe implements Strategy {

    private final Logger logger = LoggerFactory.getLogger("com.identityblitz.idp.
↪flow.dynamic");

    private final static String DOMAIN = "<BLITZ-HOST>";
    private final static String CLIENT_ID = "<CLIENT_ID>";

    @Override public StrategyBeginState begin(final Context ctx) {
        if ("login".equals(ctx.prompt())) {
            List<String> methods = new ArrayList<String>(Arrays.asList(ctx.
↪availableMethods()));
            methods.remove("cls");
            return StrategyState.MORE(methods.toArray(new String[0]), true);
        }
    }
}

```

(continues on next page)

(continued from previous page)

```

    } else {
        if(ctx.claims("subjectId") != null)
            return StrategyState.ENOUGH();
        else
            return StrategyState.MORE(new String[]{});
    }
}

@Override
public StrategyState next(Context ctx) {
    List<List<String>> choice = new ArrayList<List<String>>();
    choice.add(Arrays.asList("Value 1"));
    choice.add(Arrays.asList("Value 2"));
    try {
        if (ctx.user() == null || ctx.user().requiredFactor() == null
            || ctx.user().requiredFactor().equals(ctx.
↳justCompletedFactor())) {
            String res = new ObjectMapper().writeValueAsString(choice);
            String choiceJson = Base64.getEncoder().encodeToString(res.
↳getBytes("UTF-8"));
            return choice(ctx, choiceJson);
        }
        else
            return StrategyState.MORE(new String[]{});
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}

private StrategyState choice(final Context ctx, final String choiceJson) {
    String uri = "https://" + DOMAIN + "/blitz/pipes/choice/start?appId=" +
↳CLIENT_ID + "&pipeId=select_value&choices=" + choiceJson;
    Set<String> claims = new HashSet<String>(){
        add("instanceId");
    };
    Set<String> scopes = new HashSet<String>(){
        add("openid");
    };
    return StrategyState.ENOUGH_BUILDER()
        .withPipe(uri, CLIENT_ID, scopes, claims)
        .build();
}
}

```

### Adding a procedure to blitz.conf

in the `blitz.conf` configuration file add a section `blitz.prod.local.idp.built-in-pipes` in which assign to the auxiliary application with `choice`` type the identifier ``id specified in the procedure and the name of the attribute claim in which save the selection result.

Configuration example of the `choice` helper application:

```

"built-in-pipes": {
  "choice": [
    {
      "id": "select_value",
      "claim": "role"
    }
  ]
}

```

(continues on next page)

(continued from previous page)

```
]
}
```

### 2.4.3 Functions and methods of various purposes in login procedures

This section contains examples of functions and methods that you can use when writing Blitz Identity Provider login procedures.

#### See also:

For your convenience, Blitz Identity Provider also provides a set of [ready-made procedures](#) (page 193).

#### Obtaining the user's geodata

The login procedure can be used to obtain data about the country and city where the user is located, and based on this, flexibly configure the login rules, for example, to prohibit login from abroad, activate the second authentication factor, etc.

To do this, use the following classes and methods in the login procedures:

1. LGeoData class with `getCountry()` and `getCity()` functions.

```
public class LGeoData {
    /**
     * Get IP address country
     *
     * @return - country or null if country not specified.
     */
    public final String getCountry();

    /**
     * Get IP address city
     *
     * @return - city or null if city not specified.
     */
    public final String getCity();
}
```

2. `geoData()` method in Context.

```
/**
 * Get geo data of user IP address
 *
 * @return - geo data.
 */
LGeoData geoData();
```

**Important:** For the method to work, you need to import the LGeoData class.

```
import com.identityblitz.idp.login.authn.flow.LGeoData
```

Listing 13: An example of the code that outputs the user's country and city to the log

```
import com.identityblitz.idp.login.authn.flow.LGeoData;

LGeoData geoData = _ctx.geoData();
String country = geoData.getCountry();
logger.trace("IP location: country - {}, city - {}, factor - {}", country ,
↳geoData.getCity());
```

Listing 14: An example of a procedure involving 2FA for certain countries

```
package com.identityblitz.idp.flow.dynamic;

import java.lang.*;
import java.util.*;
import java.text.*;
import java.time.*;
import java.math.*;
import java.security.*;
import javax.crypto.*;
import org.slf4j.LoggerFactory;
import org.slf4j.Logger;
import org.codehaus.jackson.map.ObjectMapper;
import org.codehaus.jackson.type.TypeReference;
import com.identityblitz.idp.login.authn.flow.api.*;
import com.identityblitz.idp.login.authn.flow.Context;
import com.identityblitz.idp.login.authn.flow.Strategy;
import com.identityblitz.idp.login.authn.flow.StrategyState;
import com.identityblitz.idp.login.authn.flow.StrategyBeginState;
import com.identityblitz.idp.login.authn.flow.LCookie;
import com.identityblitz.idp.login.authn.flow.LUserAgent;
import com.identityblitz.idp.login.authn.flow.LBrowser;
import com.identityblitz.idp.login.authn.flow.LGeoData;
import com.identityblitz.idp.federation.matching.JsObj;
import com.identityblitz.idp.flow.common.api.*;
import com.identityblitz.idp.flow.dynamic.*;
import java.util.function.Predicate;
import java.util.stream.Stream;
import java.util.stream.Collectors;
import java.lang.invoke.LambdaMetafactory;
import java.util.function.Consumer;
import static com.identityblitz.idp.login.authn.flow.StrategyState.*;

public class EnableSecondFactorByCountry implements Strategy {

    private final Logger logger = LoggerFactory.getLogger("com.identityblitz.idp.
↳flow.dynamic");

    @Override public StrategyBeginState begin(final Context ctx) {
        if ("login".equals(ctx.prompt())){
            List<String> methods = new ArrayList<String>(Arrays.asList(ctx.
↳availableMethods()));
            methods.remove("cls");
            return StrategyState.MORE(methods.toArray(new String[0]), true);
        } else {
            if(ctx.claims("subjectId") != null)
                return StrategyState.ENOUGH();
            else
                return StrategyState.MORE(new String[]{});
        }
    }
}
```

(continues on next page)



(continued from previous page)

```

    }
}

@Override public StrategyState next(final Context ctx) {
    Integer reqFactor = (ctx.user() == null) ? null : ctx.user().
↳requiredFactor();
    LGeoData geoData = ctx.geoData();
    String country = geoData.getCountry();
    logger.info("IP location: country - {}, city - {}, factor - {}", country ,
↳geoData.getCity());
    if(ctx.justCompletedFactor() == 1 && (country == null || !country.equals(
↳"Russia")))
        return StrategyState.MORE(new String[]{});
    else
        return StrategyState.ENOUGH();
}
}
}

```

### User session reset

In a login procedure, you can force a user's session to be reset under certain conditions. To do this, use the `StrategyState.MORE_BUILDER()` function with the following methods:

- `setResetSession(reset: Boolean): true` - reset the session, `false` - do not reset (default `false`).
- `isResetSession()`: lets you know if the session has been reset.

The example below contains a script that resets a session if `ctx.prompt=login`:

```

package com.identityblitz.idp.flow.dynamic;

import java.lang.*;
import java.util.*;
import java.text.*;
import java.time.*;
import java.math.*;
import java.security.*;
import javax.crypto.*;
import org.slf4j.LoggerFactory;
import org.slf4j.Logger;
import org.codehaus.jackson.map.ObjectMapper;
import org.codehaus.jackson.type.TypeReference;
import com.identityblitz.idp.login.authn.flow.api.*;
import com.identityblitz.idp.login.authn.flow.Context;
import com.identityblitz.idp.login.authn.flow.Strategy;
import com.identityblitz.idp.login.authn.flow.StrategyState;
import com.identityblitz.idp.login.authn.flow.StrategyBeginState;
import com.identityblitz.idp.login.authn.flow.LCookie;
import com.identityblitz.idp.login.authn.flow.LUserAgent;
import com.identityblitz.idp.login.authn.flow.LBrowser;
import com.identityblitz.idp.login.authn.flow.LGeoData;
import com.identityblitz.idp.federation.matching.JsObj;
import com.identityblitz.idp.flow.common.api.*;
import com.identityblitz.idp.flow.dynamic.*;
import java.util.function.Predicate;
import java.util.stream.Stream;
import java.util.stream.Collectors;
import java.lang.invoke.LambdaMetafactory;
import java.util.function.Consumer;

```

(continues on next page)

(continued from previous page)

```

import static com.identityblitz.idp.login.authn.flow.StrategyState.*;

public class ResetSession implements Strategy {

    private final Logger logger = LoggerFactory.getLogger("com.identityblitz.idp.
↪flow.dynamic");

    @Override public StrategyBeginState begin(final Context ctx) {
        if ("login".equals(ctx.prompt())){
            List<String> methods = new ArrayList<String>(Arrays.asList(ctx.
↪availableMethods()));
            methods.remove("cls");
            logger.info("### RESET_SESSION");
            return StrategyState.MORE_BUILDER().setResetSession(true).
↪addMethods(methods.toArray(new String[0])).build();
        } else {
            if(ctx.claims("subjectId") != null)
                return StrategyState.ENOUGH();
            else
                return StrategyState.MORE(new String[]{});
        }
    }

    @Override public StrategyState next(final Context ctx) {
        Integer reqFactor = (ctx.user() == null) ? null : ctx.user().
↪requiredFactor();
        if(reqFactor == null || reqFactor == ctx.justCompletedFactor())
            return StrategyState.ENOUGH();
        else
            return StrategyState.MORE(new String[]{});
    }
}

```

### Invoking custom errors in script

Blitz Identity Provider allows you to create custom errors and call them in login procedures. Do the following:

1. Following the [instructions](#) (page 234), add a custom error message to the messages file in the /usr/share/identityblitz/blitz-config/custom\_messages directory.

```
err.bad_gateway=Недоступно
```

2. Call this error upon getting HTTP 502.

```

if (result.status() == 502) {
    return HttpLoop.error("bad_gateway",
        Collections.<String, String>
↪singletonMap("status", "" + result.status()));
}

```

Sample script that calls a custom HTTP 502 error for the [Flash Call](#) (page 104) authentication method:

```

package flashcall;

import com.identityblitz.core.loop.http.HttpLoop;
import com.identityblitz.core.loop.http.HttpLoopRequest;
import com.identityblitz.core.loop.http.HttpLoopResult;
import com.identityblitz.core.loop.JsObj;

```

(continues on next page)

(continued from previous page)

```

import java.util.Collections;

public class FlashCallFlow implements HttpLoop {

    public HttpLoopRequest run(final JsObj obj, final HttpLoopResult_
↪result) {
        if (result == null) {
            final String number = obj.asString("phone_number");
            return HttpLoop.callBuilder("POST", "http://test.
↪flashcall.ru/api/v1")
                .withHeader("Token", "1234567890")
                .withBody(JsObj.empty.addString("id",
↪"test").addString("dst_number", number.substring(number.length() - 10)))
                .build(JsObj.empty);
        } else if (result.status() == 200) {
            final JsObj body = result.body();
            return HttpLoop.Ok(JsObj.empty.addString("code", body.
↪asString("SenderID")));
        } else if (result.status() == 502) {
            return HttpLoop.error("bad_gateway",
                Collections.<String, String>
↪singletonMap("status", "" + result.status()));
        } else {
            return HttpLoop.error("wrong_http_status",
                Collections.<String, String>
↪singletonMap("status", "" + result.status()));
        }
    }
}

```

### Analyzing application tags

Blitz Identity Provider allows you to assign [tags](#) (page 157) to applications and set the operation logic regarding the tagged apps in login procedures.

To retrieve application tags, a procedure must use the `ctx.appTags()` method within `Context`.

**Attention:** For the method to work, you must import `java.util.Set`.

An example of a procedure that obtains the 2F tag and uses it to enable the second factor authentication:

```

package com.identityblitz.idp.flow.dynamic;

import java.lang.*;
import java.util.*;
import java.text.*;
import java.time.*;
import java.math.*;
import java.security.*;
import javax.crypto.*;
import org.slf4j.LoggerFactory;
import org.slf4j.Logger;
import org.codehaus.jackson.map.ObjectMapper;
import org.codehaus.jackson.type.TypeReference;
import com.identityblitz.idp.login.authn.flow.api.*;
import com.identityblitz.idp.login.authn.flow.Context;
import com.identityblitz.idp.login.authn.flow.Strategy;

```

(continues on next page)

(continued from previous page)

```

import com.identityblitz.idp.login.authn.flow.StrategyState;
import com.identityblitz.idp.login.authn.flow.StrategyBeginState;
import com.identityblitz.idp.login.authn.flow.LCookie;
import com.identityblitz.idp.login.authn.flow.LUserAgent;
import com.identityblitz.idp.login.authn.flow.LBrowser;
import com.identityblitz.idp.login.authn.flow.LGeoData;
import com.identityblitz.idp.federation.matching.JsObj;
import com.identityblitz.idp.flow.common.api.*;
import com.identityblitz.idp.flow.dynamic.*;
import java.util.function.Predicate;
import java.util.stream.Stream;
import java.util.stream.Collectors;
import java.lang.invoke.LambdaMetafactory;
import java.util.function.Consumer;
import static com.identityblitz.idp.login.authn.flow.StrategyState.*;

public class UseAppTags implements Strategy {

    private final Logger logger = LoggerFactory.getLogger("com.identityblitz.idp.
↪flow.dynamic");

    @Override public StrategyBeginState begin(final Context ctx) {
        if ("login".equals(ctx.prompt())){
            List<String> methods = new ArrayList<String>(Arrays.asList(ctx.
↪availableMethods()));
            methods.remove("cls");
            return StrategyState.MORE(methods.toArray(new String[0]), true);
        } else {
            if(ctx.claims("subjectId") != null)
                return StrategyState.ENOUGH();
            else
                return StrategyState.MORE(new String[]{});
        }
    }

    @Override public StrategyState next(final Context ctx) {
        Set<String> tags = ctx.appTags();
        logger.info("APP TAGS: " + tags);
        if (ctx.justCompletedFactor() == 1 && tags.contains("2F"))
            return StrategyState.MORE(new String[]{});
        else
            return StrategyState.ENOUGH();
    }
}

```

## 2.4.4 Customization of the logic of operations with data storages

### Customization principle

Blitz Identity Provider allows you to customize the logic of operations with data storages. To do this, a Java class with a fixed name and the package `com.identityblitz.idp.store.id.logic.dynamic` is used.

There are eight custom procedures, one for each operation with a fixed class name:

- `searchUser` — `CustomSearchUsersLogic.java`
- `getUser` — `CustomGetUserLogic.java`
- `findUser` — `CustomFindUserLogic.java`
- `bindUser` — `CustomBindUserLogic.java`

- `changeUserPassword` — `CustomChangeUserPasswordLogic.java`
- `addUser` — `CustomAddUserLogic.java`
- `updateUser` — `CustomUpdateUserLogic.java`
- `deleteUser` — `CustomDeleteUserLogic.java`

## Configuration

To configure custom logic for the required operations, follow these steps:

1. Place Java files with custom logic in a directory:

```
/usr/share/identityblitz/blitz-config/dynamic/idstore/<operation_name_in_
↳lowercase>
```

For example, to enable custom logic for `searchUsers` and `bindUser`, place the files `CustomSearchUsersLogic.java` and `CustomBindUserLogic.java` to the directories `/usr/share/identityblitz/blitz-config/dynamic/idstore/searchusers` and `/usr/share/identityblitz/blitz-config/dynamic/idstore/binduser` respectively.

2. Open the configuration file `/usr/share/identityblitz/blitz-config/blitz.conf`.

```
sudo vim /usr/share/identityblitz/blitz-config/blitz.conf
```

3. Add a new logic block to the `blitz.prod.local.idp.id-stores` block. The new block must contain the names of the customized operations specified as the key and the `{ "enabled": true}` section as the key value.

Listing 15: Customization of `searchUsers` and `bindUser` operations

```
{
  "logic": {
    "searchUsers": {
      "enabled": true
    },
    "bindUser": {
      "enabled": true
    }
  }
}
```

## Writing a custom procedure

Custom procedures for all operations have the same specification, but use their own context and utility functions. Each method in the procedures corresponds to a specific state of the operation execution process. In the methods, it is necessary to implement the logic of moving to the next cycle (followed by calling a new method) or completing the operation.

Each method in the procedure returns a pair of `LoopOutput` and `OperationState`. `LoopOutput` can be:

1. `terminal` – completes the logical cycle of operation in one of the following ways:
  - `error`;
  - `success` (the result of success for a certain operation);
  - the final save operation (perform the save operation with some parameters and finish with the result).
2. `task` - more cycle iterations are required:
  - request to the repository to perform a specific operation;

- request to an external web service.

At the moment, the mechanism of custom procedures is being beta tested. You can request a detailed Java specification and get advice on customization options in your environment from our technical specialists at [support@idblitz.ru](mailto:support@idblitz.ru).

### 2.4.5 Procedures for binding external user accounts

Besides the *basic configuration* (page 122), it is possible to bind accounts for each external identity provider by using a binding procedure in Java. This mode provides maximum configuration flexibility and is suitable for highly specialized account binding and attribute mapping scenarios.

The customization is available under Identity providers -> Account linking -> Advanced customization. To write your own procedure, follow the instructions in the basic procedure as well as the recommendations in this section of the documentation.

## Account linking

Basic configuration

Advanced configuration

## Account linking procedure

For the binding procedure to work successfully, you must write a class in `Java` that inherits the abstract class `MatchingBlock`. The class name must be `Google_IGoogle`. The class must have a public `default` constructor. For security purposes, the class is loaded by a separate `class loader` with a limited list of `imports`. All necessary information is passed to function parameters.

```

1 package com.identityblitz.idp.federation.matching.dynamic;
2
3 import java.lang.*;
4 import java.util.*;
5 import java.text.*;
6 import java.time.*;
7 import java.math.*;
8 import java.security.*;
9 import javax.crypto.*;
10 import org.slf4j.LoggerFactory;
11 import org.slf4j.Logger;
12 import com.identityblitz.idp.federation.*;
13 import com.identityblitz.idp.federation.matching.*;
14 import com.identityblitz.idp.flow.common.api.*;
15 import com.identityblitz.idp.flow.common.model.*;
16 import com.identityblitz.idp.federation.matching.dynamic.*;
17 import java.util.function.Consumer;
18 import java.util.stream.Stream;
19 import java.util.stream.Collectors;
20 import org.codehaus.jackson.map.ObjectMapper;
21 import org.codehaus.jackson.type.TypeReference;
22 import com.identityblitz.idp.extensions.types.JsonObject;
23
24 import com.identityblitz.idp.federation.matching.*;
25 import com.identityblitz.idp.flow.common.api.HttpFactory;
26
27 /**
28  * The class is inherited from MatchingBlock and must have a default constructor to be instantiated correctly.
29  * The current generated implementation provides a strategy where users are not matched or updated.
30  */
31 public class Google_IGoogle extends MatchingBlock {
32
33     private final Logger logger = LoggerFactory.getLogger("com.identityblitz.idp.federation.matching.dynamic");
34
35     /**
36      * Iterative function determining the correspondence of internal accounts and identity provider accounts.
37      * At each iteration, the function can perform a find operation (found users will be passed in the next iteration)
38      * or terminate the operation with the following solutions:
39      * matched - matching users found;
40      * matchError - user matching error;
41      * matchByLogin - connect to a user who has successfully authenticated;
42      * refine - receives a list of users, asks for the password and connects with the user who entered the correct password;
43      * @param ctx - procedure context with the following fields:
44      *   iteration - procedure iteration number;
45      *   extAttrs - user attributes received from the identity provider;
46      *   sid - unique identifier of the external account.
47      * @param users - users.
48      * @return - one of the listed solutions
49      */
50     @Override public MatchResult match(MatchingContext ctx, List<MatchingUser> users){
51         return matchError(ctx, new MatchingError("not_matched", "User not matched"));
52     };
53
54     /**
55      * Returns attributes which can be updated or deleted.
56      * @param extAttrs - user attributes received from the identity provider.
57      * @param user - internal user.
58      * @param justMatched - an indication that the linking of internal accounts with external vendor accounts is established for the first time.
59      * @return - tuple with changeable and deleteable attributes. For example: change(JsonObj.empty(), Collections.<String>emptySet())
60      */
61     @Override public Tuple2<JsonObject, Set<String>> update(JsonObj extAttrs, MatchingUser user, Boolean justMatched, HttpFactory httpFactory){
62         return change(JsonObj.empty(), Collections.<String>emptySet());
63     };
64 }
65

```

Cancel

Save

## User registration in external identity provider

In the form where you enter login and password to authenticate through an external identity provider, you may see a link to the external provider registration page (No account? Register). In order for the link not to be displayed, the `refine` and `matchByLogin` functions in the `bind` procedure must be called without specifying registration parameters.

- `refine(cxt, users)` instead of `refine(cxt, users, regAttrs)`;
- `matchByLogin(cxt)` instead of `matchByLogin(cxt, regAttrs)`.

Here's an example of how to use those functions in a procedure:

```
package com.identityblitz.idp.federation.matching.dynamic;

import java.lang.*;
import java.util.*;
import java.text.*;
import java.time.*;
import java.math.*;
import java.security.*;
import javax.crypto.*;
import org.slf4j.LoggerFactory;
import org.slf4j.Logger;
import com.identityblitz.idp.federation.*;
import com.identityblitz.idp.federation.matching.*;
import com.identityblitz.idp.flow.common.api.*;
import com.identityblitz.idp.flow.common.model.*;
import com.identityblitz.idp.federation.matching.dynamic.*;
import java.util.function.Consumer;
import java.util.stream.Stream;
import java.util.stream.Collectors;
import org.codehaus.jackson.map.ObjectMapper;
import org.codehaus.jackson.type.TypeReference;
import com.identityblitz.idp.extensions.types.JsonObject;
import com.identityblitz.idp.federation.matching.*;
import com.identityblitz.idp.flow.common.api.HttpFactory;

public class Esia_1Esia extends MatchingBlock {

    private final Logger logger = LoggerFactory.getLogger("com.identityblitz.idp.
↵federation.matching.dynamic");

    @Override public MatchResult match(MatchingContext ctx, List<MatchingUser> users)
↵{
        if (ctx.iteration() == 1) {
            return find(ctx, MatchingFilter.empty().eq("uid", "00000").or().eq("uid",
↵"test@test.ru"));
        } else {
            //return refine(ctx, Collections.singletonList((users.get(0))));
            //return refine(ctx, users);
            return matchByLogin(ctx);
        }
    };

    @Override public Tuple2<JsonObject, Set<String>> update(JsonObject extAttrs, MatchingUser
↵user, Boolean justMatched, HttpFactory httpFactory){
        return change(JsonObject.empty(), Collections.<String>emptySet());
    };
}
```



## Discovering external account name

A bind procedure allows you to discover the name of a user external account and update the relevant parameter in the database each time the user logs in through an external identity provider. To do so, use the `updateFederatedAccountName` function.

Here's an example of how to use the function in a procedure:

```
package com.identityblitz.idp.federation.matching.dynamic;

import java.lang.*;
import java.util.*;
import java.text.*;
import java.time.*;
import java.math.*;
import java.security.*;
import javax.crypto.*;
import org.slf4j.LoggerFactory;
import org.slf4j.Logger;
import com.identityblitz.idp.federation.*;
import com.identityblitz.idp.federation.matching.*;
import com.identityblitz.idp.flow.common.api.*;
import com.identityblitz.idp.flow.common.model.*;
import com.identityblitz.idp.federation.matching.dynamic.*;
import java.util.function.Consumer;
import java.util.stream.Stream;
import java.util.stream.Collectors;
import org.codehaus.jackson.map.ObjectMapper;
import org.codehaus.jackson.type.TypeReference;
import com.identityblitz.idp.extensions.types.JsObject;

import com.identityblitz.idp.federation.matching.*;
import com.identityblitz.idp.flow.common.api.HttpFactory;

public class Esia_1Esia extends MatchingBlock {

    private final Logger logger = LoggerFactory.getLogger("com.identityblitz.idp.
↵ federation.matching.dynamic");

    @Override public MatchResult match(MatchingContext ctx, List<MatchingUser> users)
↵ {

        if (ctx.iteration() == 1) {
            //return matchError(ctx, new MatchingError("bad_err_code", "bad_err_msg"));
            return tryToSearch(ctx);
        }

        if (users.isEmpty()) {
            return matchError(ctx, new MatchingError("error", "error"));
        }

        if (users.size() == 1) {
            return matched(ctx, users.get(0));
        }

        return refine(ctx, users, ctx.extAttrs());
    };

    private MatchResult tryToSearch(MatchingContext ctx) {
        return find(ctx, filter().eq("uid", "test@test.ru"));
    }
}
```

(continues on next page)

(continued from previous page)

```

@Override public Tuple2<JsObj, Set<String>> update(JsObj extAttrs, MatchingUser_
↪user, Boolean justMatched, HttpFactory httpFactory){
    return change(JsObj.empty(), Collections.<String>emptySet());
};

@Override public boolean isAllowMultiBind() {
    return true;
}

@Override public String updateFederatedAccountName(JsObj extAttrs){
    if (extAttrs.contains("firstName") && extAttrs.contains("lastName")){
        String name = extAttrs.asString("firstName") + " " + extAttrs.asString(
↪"lastName");
        if (extAttrs.contains("middleName")) {
            name = name + " " + extAttrs.asString("middleName");
        }
        return name;
    } else {
        // don't update federated account name
        return super.updateFederatedAccountName(extAttrs);
    }
};
}

```

## 2.5 Design and UI texts

### 2.5.1 Login page

**Warning:** The administrator of the admin console must personally check if JS-scripts placed on the login page are correct and make sure that content of the login page is free of vulnerabilities.

In the *“Login page themes”* section of the Admin Console, the administrator can customize the appearance settings for the single sign-on page. If Blitz Identity Provider user registration and password recovery applications are used, their appearance will also match the settings for the single sign-on page appearance.

When you enter the *“Appearance”* section, a list of customized login page templates is displayed. Each template is described by:

- template identifier;
- template name;
- applications list;
- description.

By default, a template with the `default` identifier is created - this is used for all applications connected to Blitz Identity Provider, as well as for single logout pages.

The default template is edited using a special constructor (more details below).

Also you can:

- create and modify new templates using the builder and assign them to different applications;
- create and modify new templates manually.

## Editing the default template

When opening the default template editing page the following information is displayed about the template itself (template identifier, template name, description and applications list), as well as the interface of the login page template builder.

Customizing the appearance of the login page (template properties):

**Template properties**

Template identifier:

Template name:

Тэги шаблона:   
Для default шаблона задать тэги нельзя

Description:

Applications:

[Save](#)

Customizing the appearance of the login page (login page appearance):

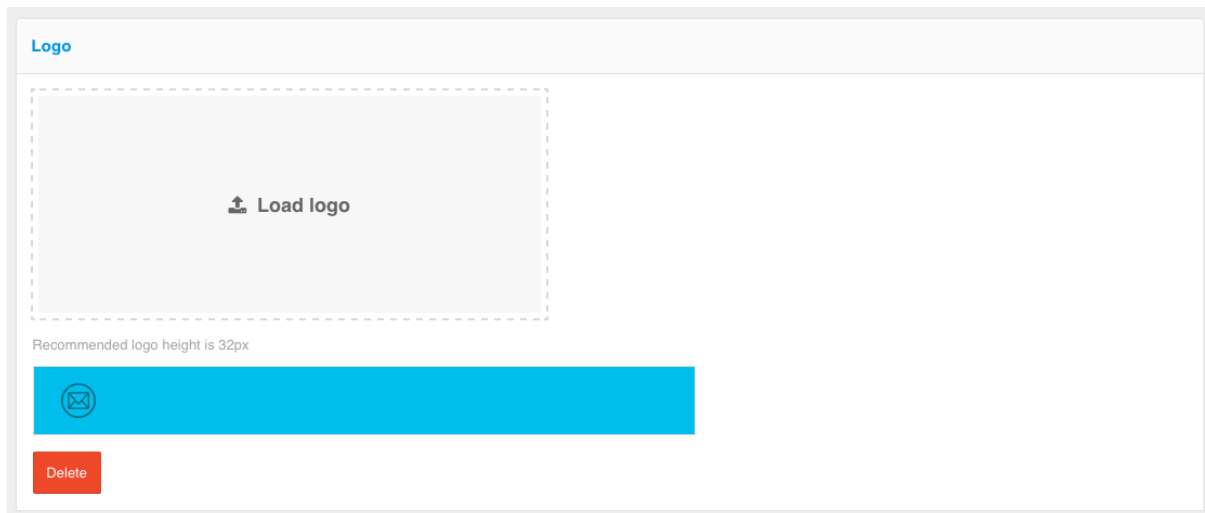
**Login page customization**

Theme:

Alignment of the main login form:
  Left
  Center
  Right

Select language:

Customizing the appearance of the login page (logo):



Customizing the appearance of the login page (background image):




**Background image**


📁 Load background image

Recommended file size is less than 1MB

Or choose one from the images

Image 1     Image 2     Image 3





After deleting the background image a default image will be used.

[Delete](#)

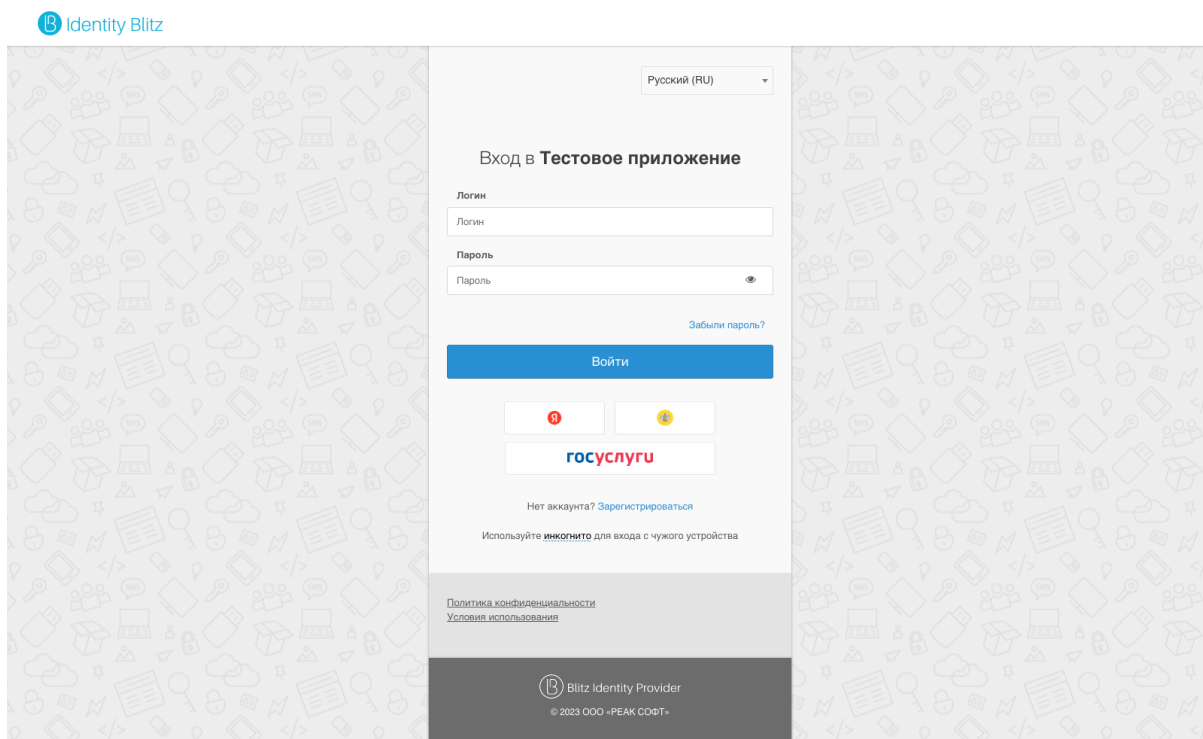
Customizing the appearance of the login page (customizing the footer):

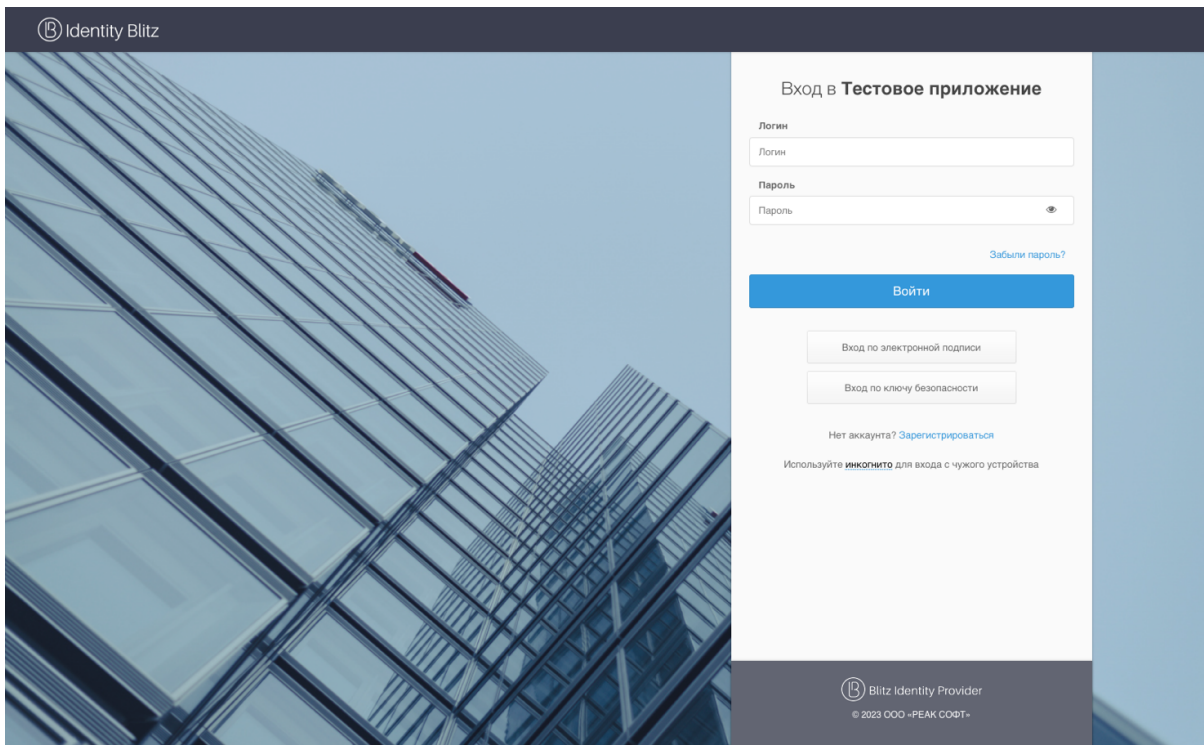


In standard configuration Blitz Identity Provider provides the following features:

- three color themes for the interface elements;
- ability to define the location of the main login form block (identification and authentication, registration, password recovery);
- ability to upload a company logo to be displayed in the page header;
- choice of a background image (you can choose from 3 standard images in each theme, or you can upload your own, custom background image);
- login page footer properties.

The figures below show some examples of login pages resulting from the default configuration.










### Creating and modifying new templates using the constructor

Blitz Identity Provider allows you to customize different login pages for the case when a user logs into different connected applications. To do this, you need to create new login templates - the easiest way to do this is to do it on the basis of an existing default-template by clicking the “Copy” button. After that a new template will be created, which can be edited using the constructor.

**Login page templates**

Use templates to design the login page. You can edit the master template or create new templates that will be used for pages. You can create additional templates by saving a copy of the main template and manually edit the templates.

Identifier	Template name	Applications	Description	
default	Default theme	oauth_test	Generated at 1482151154	 
default_1482928471 	Default theme		Generated at 1482151154	 

In order for the new template to be used when entering a certain application, you should go to the “Applications” section to edit the required application and select the required page template.

**Application settings**

Identifier (entityID or client\_id)   
Application identifier. Used for identifying application within the SAML (corresponds to entityID) and OAuth 2.0 (corresponds to client\_id) protocols.

Name   
Human-readable application name. Is used only inside Blitz Identity Provider.

Domain   
Usually a link to the application's start page, e.g. http://testdomain.ru/. If TLS-authentication is used, then the domain should correspond to the domain specified in the certificate.

Application start page   
Link to the application start page, e.g. http://testdomain.com/private. When logging in using SAML, it is used as a link to go to the application in case the login page is opened from the browser history

Identifier encryption key   
If the key is specified, the user ID for the application will be encrypted using this key. The key value can be selected from a list. You can also assign a new key by typing it in the search box and pressing Enter

**Page template**   
Page template determines the login page appearance. If the template is not specified, then the default template is used.

Logout redirect uri prefixes   
A prefix is used to check the redirect uri. If the logout request includes an post\_logout\_redirect\_uri that doesn't correspond to any prefix, then the logout is rejected

### Creating and modifying new templates in manual mode

You can customize the appearance of the login page to meet your organization's individual requirements, i.e. there is no need to be limited to the features of the builder only.

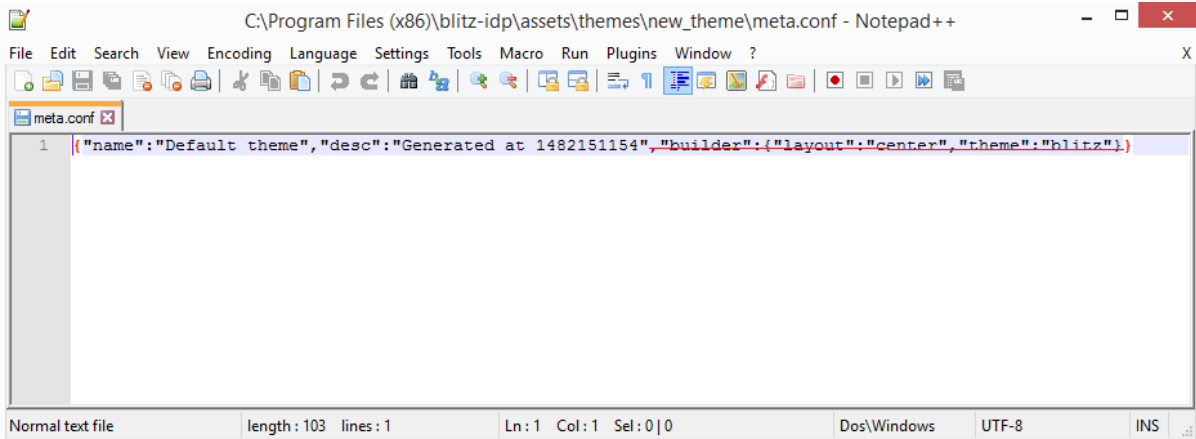
Each template of the login page is a zip archive. Each login page template is a zip archive. All templates are placed in the directory:

```
\assets\themes
```

The easiest way to manually edit the template is to take the following steps:

- create a copy of an existing template (e.g. default-template) by clicking the "Copy" button in the console;
- go to the template directory;
- unpack the archive with the new created template;
- edit the `meta.conf` file contained in the archive by removing the `builder` parameter;
- zip the template files back up, making sure the `meta.conf` file is in the root directory.





After completing these steps, you will be able to edit the theme manually. In addition to the standard fields describing the theme itself, the “*Page template*” block is available. It allows you to create / modify a template - a text file that is compiled using the Twirl template engine <<https://www.playframework.com/documentation/2.5.x/ScalaTemplates>>.

The template must have a signature:

```
@(headers: Html, fBuilder: FormBuilder, scripts: Html, path: String)(implicit_
  ↳ request: RelyingPartyRequest[_], messages: Messages)
```

You should use following parameters when creating a template:

- `headers` is the HTML code for the page title, which should be placed in the `head` tag;
- `form` - HTML code of the main form, which should be placed in the `body` tag;
- `scripts` - HTML code with JavaScript required for the form to work correctly, which should be placed in the `body` tag;
- `pathAssets` - context path to template resources.

The `@fBuilder()` function adds the code of the main authentication form to the page. The authentication form (list and composition of fields, location of buttons) is not customizable except for changes implemented by CSS means. In other words, CSS tools can be used to change the color of individual elements or hide them - to do this, find the corresponding class in the theme’s CSS file and change its properties.

An example of the basic template is shown below:

```
@(headers: Html, fBuilder: FormBuilder, scripts: Html, path: String)(implicit_
  ↳ request: RelyingPartyRequest[_], messages: Messages)

<!DOCTYPE html>
<html>

<head>
  @headers
</head>

<body>
  <div id="main">
    <section id="content_wrapper">
      @fBuilder()
    </section>
    <div>
      <div>
        @Html(messages("author.copyright"))
      </div>
```

(continues on next page)

(continued from previous page)


```
</div>
</div>
@scripts
</body>

</html>
```

When using this template, the login page will look like the one shown in the figure below.

Log in to **User profile**

**Username**

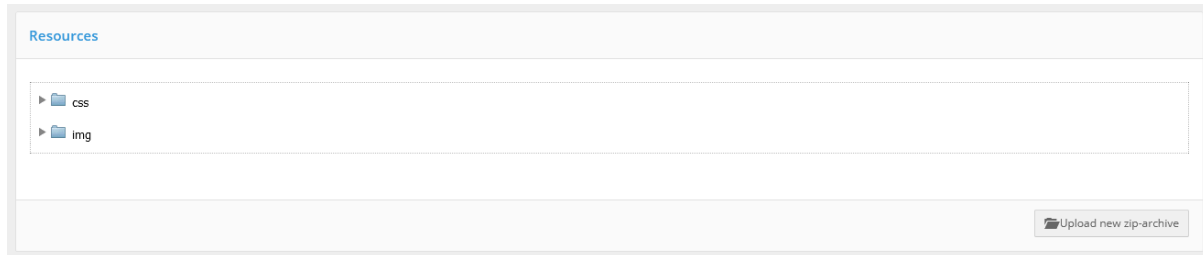
  
**Password**   
[Forgot password?](#)

**Sign in**

[No account? Register](#)

When you design a login page template you have the ability to use resources, like CSS and images.

To upload them, you should use the “Resources” block of the page appearance, which allows you to upload the necessary files in a zip archive. To make the corresponding files available, they should be placed in the archive directory named `assets`. The required resources can also be manually included in the original zip archive with the page template.



To enable a language switch in the template body, add the following block:

```
<div ...>
  <section class="language-section">
    <div class="language-selector">
      <select id="lang-selector"></select>
    </div>
  </section>
  @langSelector ()
</div>
```

## 2.5.2 User profile

Blitz Identity Provider allows you to change the header and footer logos in the User profile, as well as customize the User profile color scheme using CSS.

### Header logo

To replace the header logo of the User profile, use one of the following methods:

#### Method #1

Replace the `logo-ib_h30.png` logo file in the `.../assets/public/lib/blitz-common/` directory with a new file of the same name.

#### Method #2

1. Put a file with the custom logo (`mylogo.png` in the example below) into the `.../assets/public/lib/blitz-common/` directory.
2. Open the `.../assets/public/lib/blitz-profile/stylesheets/custom.min.css` file.

```
sudo vim /usr/share/identityblitz/blitz-config/assets/public/lib/blitz-profile/
→stylesheets/custom.min.css
```

3. Specify the URL path to the new file.

**Attention:** For the path to the `.../assets/public/lib/blitz-common/` directory, use `https://<domain> /blitz/assets/img/`.

```

:root{
  --navbar-branding-img: url(https://<domain>/blitz/login/assets/img/mylogo.
  ↳png);
  ...
}

```

**Tip:** For the logo, you can also use a file available on a public URL.

### Footer logo

To change the logo in the footer of the User profile, replace the `logo-bip.png` logo file in the `.../assets/public/lib/blitz-common/` directory with a new file of the same name.

### Color scheme customization

Changing the color scheme is conducted in the `.../assets/public/lib/blitz-profile/stylesheets/custom.min.css` file.

```

:root{
  --profile-color-accent:#00bde5;
  --profile-color-border-primary:#ddd;
  --profile-color-border:#ddd;
  --profile-color-button:#f1f1f1;
  --profile-color-href-hover:#1d6fa5;
  --profile-color-href:#3498db;
  --profile-color-outline:#ddd;
  --profile-color-primary:#3498db;
  --profile-color-text-button:#666;
  --profile-color-text-light:#fff;
  --profile-color-text-primary:#3498db;
}

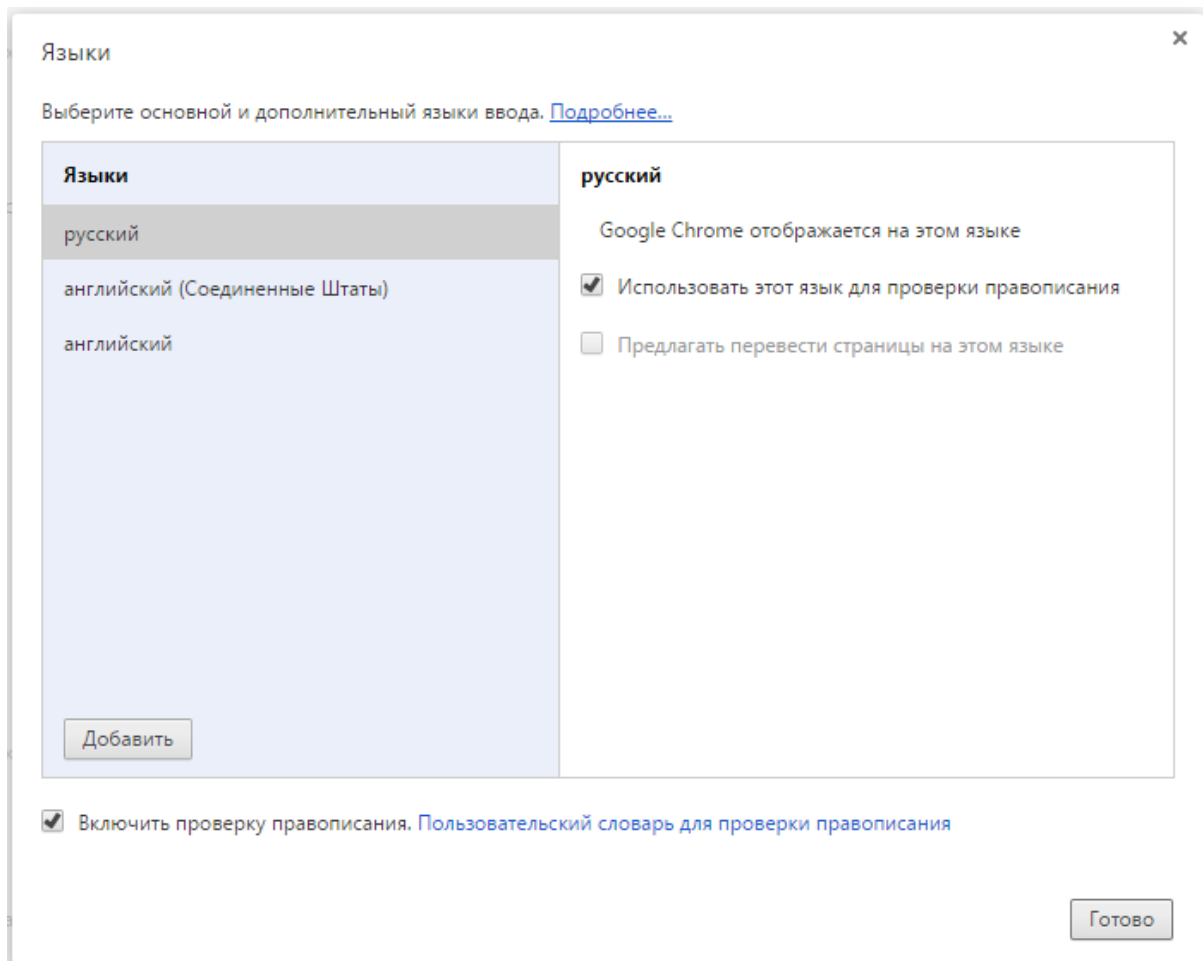
```

## 2.5.3 Multilanguage support

Blitz Identity Provider web interface supports multi-language. Two languages are provided by default - Russian and English.

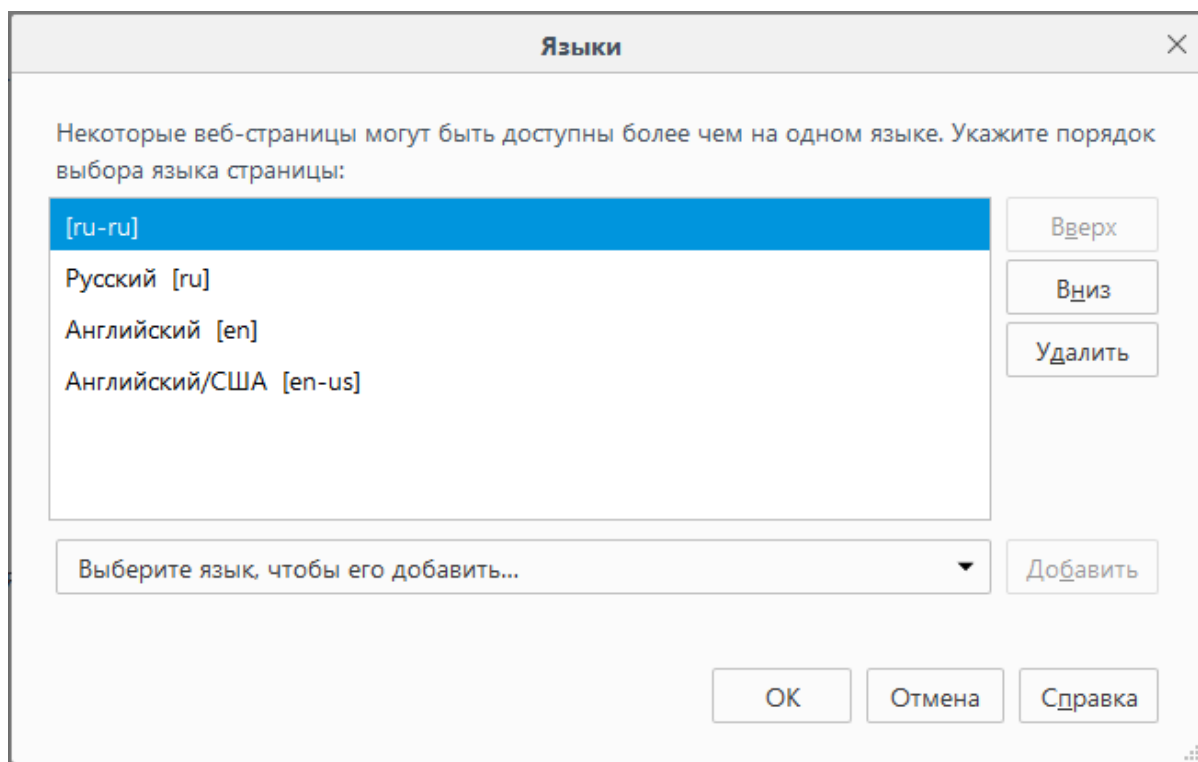
By default, the interface is displayed to the user in the language that corresponds to their system language in the OS and their preferred language in the browser. In this case, you can switch the language by changing the primary input language (the language in which web pages are displayed) in the browser you are using. For example, to change the language in the Chrome browser, follow the steps:

- go to the browser settings (`chrome://settings/`);
- select Show additional settings;
- click on the Change language preferences button;
- move the desired language to the first place in the list.



To change the language in Firefox browser, you need to follow the steps:

- go to the browser settings (about : preferences);
- select the General section of the settings;
- in the Languages subsection, click on the Select button;
- move the desired language to the first place in the list.



Additionally, it is possible to configure the language using the `blitz.conf` configuration file. To do this, edit the language setting section `blitz.prod.local.idp.lang` with the following parameters:

- `languages` – list of available languages. The first language in the list is considered to be the default language;
- `portal-lang-cookie` – name (name) and setting domain (domain) of the cookie with the current portal language (optional). If a portal cookie is set, the language change in Blitz Identity Provider is stored in the specified cookie;
- `ignore-browser` – whether or not the browser language ignore mode is turned off;
- `lang-variants` – *list of identifiers for special sets of strings for individual applications* (page 241).

The example of configuration file excerpt:

```
"lang" : {
  "ignore-browser" : true,
  "languages" : [
    "ru",
    "en"
  ],
  "lang-variants": ["special1", "special2"],
  "portal-lang-cookie" : {
    "domain" : "domain.com",
    "name" : "blitzlng"
  }
}
```

Thus, for example, if the use of the English interface language is not required, it can be removed from the `languages` parameter.

## 2.5.4 Interface text settings

### Web interface texts

Blitz Identity Provider allows you to change text strings used in the system interface. To do this, you need to edit the `messages` file located in the `/custom_messages/` directory by adding a string like “parameter=value”, where `parameter` is the text string identifier and `value` is the required text.

All text strings used by Blitz Identity Provider by default are saved in the `messages.zip` archive included with the software.

For example, the following string is responsible for the text in the registration form that contains URL to the User agreement:

```
reg.page.reg.action.agreement=By clicking  &laquo;Register&raquo; you 
↪agree with the c <a href="{0}" target="_blank">Terms of Use</a>
```

The file must be saved in UTF-8 encoding in order to display correctly.

If you need to change the English language, add the `messages.en` file to the specified directory and change the necessary files in it.

If you want to use the `@` character in texts, you must enter it twice.

### Email and SMS templates

Email templates are text strings saved in the same way as regular strings in the web interface. They are modified in the same way.

The unified format of message codes is used, which has the following form:

```
message. ${ группа_сообщений } . ${ тип_сообщения } . ${ вариация } . ${ канал } . ${ часть }
```

Following message groups are used:

- `notif` - for notifications;
- `reg` - for interaction with the user during registration;
- `recovery` - for interaction with the user when restoring access;
- `auth` - for interaction with the user during authentication;
- `profile` - for interaction with the user in the User profile;
- `api` - for interaction with the user when using API.

Message types from different groups:

#### notif

##### login\_unknown\_device

User notification about the login from unknown device.

Parameters:

- `device` - code of the device;
- `device.msg` - name of the device computed with `msg(audit.device. ${ device} )`;
- `browser` - user browser;
- user session attributes;

- `ua.name` - device name;
- `app.id` - application identifier;
- `app.name` - application name;
- `ip` - IP-address;
- `ip.country` - country;
- `ip.state` - region;
- `ip.city` - city;
- `ip.lat` - latitude;
- `ip.lng` - longitude;
- `ip.radius` - radius of the neighborhood;
- `device.type` - device type;
- `device.mkey` - collected key for messages, formation rule: `s"$deviceType.$osName.$osVer"`;
- `os.name` - operating system name;
- `os.ver` - operating system version;
- `os.mkey` - collected key for messages, formation rule: `s"$osName.$osVer"`;
- `event.time` is the time of the event (in `unixtime`).

You can use the following formatting features in a message template:

- `$(<ATTR>&dic(<MSG_KEY_PREFIX>, <PARAM_SUBSTITUTION>)]` - get value from string;
- `$(<ATTR>&formatUnixTime(dd MMMM YYYY year, ru, GMT)]` - date and time formatting, where `dd MMMM YYYY` - template in `SimpleDateFormat` format, `ru` - locale (optional), `GMT` - time-zone (optional).

In the template, you can set conditions for the presence of parameters. The following example allows you to display the word `City` and the value from the parameter `ip.city` if available, if `ip.city` is missing, then nothing will be shown:

```
$(ip.city+Город: ]$(ip.city-]
```

---

**Tip:** For the example to work, create and activate the login procedure [extracting user's geodata](#) (page 210).

---

### link\_social\_network

User notification about linking to social network.

Parameters:

- `fp.humanReadableName` - name of the external identity provider;
- user attributes.



### change\_pwd

User notification about password change.

Parameters:

- user attributes.

### changed\_pwd\_to\_object

User notification about password change in dependent account.

Parameters:

- attributes of the dependent account with `obj` prefix.

### access\_recovery

User notification about password recovery

Parameters:

- user attributes.

### access\_recovery\_by\_object

User notification about password recovery in dependent account.

Parameters:

- attributes of the dependent account with `obj` prefix.

### set\_2factor\_auth

User notification of the assignment of the second authentication factor.

Parameters:

- `method` - authentication method code;
- `method.msg` - authentication method name computed by the `msg(message.method.name.[method])` string;
- user attributes.

### granted\_access\_to

Subject notification about granted access to the object.

Parameters:

- `blitz_right` - access rights code;
- subject attributes;
- object attributes with the `obj` prefix.

### granted\_access\_on

Object notification about granted access to it.

Parameters:

- `blitz_right` - access rights code;
- subject attributes;
- object attributes with the `obj` prefix.

### revoked\_access\_to

Subject notification about revoked access to the object.

Parameters:

- `blitz_right` - access rights code;
- subject attributes;
- object attributes with the `obj` prefix.

### revoked\_access\_on

Object notification about revoked access to it.

Parameters:

- `blitz_right` - access rights code;
- subject attributes;
- object attributes with the `obj` prefix.

### on\_registration

User notification about registration of his/her account.

Parameters:

- `_entryPoint_` - registration channel;
- `_appId_` - application;
- `_requesterId_` - application;
- user attributes.

Example line:

```
message.notif.login_unknown_device.email.body=Уважаемый пользователь!<br><br>Мы
↪обнаружили, что вы вошли в систему с нового устройства ${event.time&
↪formatUnixTime(dd MMMM YYYY г., ru, GMT)}:<br>${device.mkey&dic(dics.devices, os.
↪ver)}, браузер ${ua.name&dic(dics.browsers)}<br>Если вы не совершали это
↪действие, обратитесь к администратору.
```

## reg

### vrf\_code

Sending contact confirmation code during registration.

Parameters:

- `code` – confirmation code;
- `link` – confirmation link (only for `email` channel);
- `req.ip` – IP-address;
- `req.userAgent` - `userAgent` of the user;
- `cfg.domain` - domain;
- user attributes from the registration context with the prefix `attrs`.

### set\_pwd\_link

Sending the link to change password during registration (only for `email` channel).

Parameters:

- `link` - link to password change page;
- `req.ip` – IP-address;
- `req.userAgent` - `userAgent` of the user;
- `cfg.domain` - domain;
- user attributes from the registration context with the prefix `attrs`.

### generated\_pwd

Sending the assigned registration password (only for `SMS` channel).

Parameters:

- `pwd` - generated password;
- `req.ip` – IP-address;
- `req.userAgent` - `userAgent` of the user;
- `cfg.domain` - domain attributes of the user from the registration context with the prefix `attrs`.

## recovery

### vrf\_code

Sending contact confirmation code during access recovery.

Parameters:

- `code` – confirmation code;
- `link` – confirmation link (only for `email` channel).

## auth

### vrf\_code

Sending mobile number confirmation code (channels: `SMS/push`).

Parameters:

- `code` – confirmation code.

## profile

### vrf\_code

Sending confirmation code if it was changed in User profile.

Parameters:

- `attr.msg` - name of the attribute in the profile form;
- `attr` – attribute code;
- `link` – confirmation link (only for `email` channel);
- `code` – confirmation code.

## api

### vrf\_code

Variations:

- `attr.$rpId` – separately for current application and attribute;
- `attr` - separately for this attribute.

Sending contact confirmation code via API

Parameters:

- `code` – confirmation code;
- `link` – confirmation link (only for `email` channel);
- `attr.value` - new contact (e-mail or cell phone);
- `attr` – contact attribute code.

Variations allow you to specify variations in addition to the basic message template (for example, a separate template by application). The presence of a variation is checked by the basic template with the message text (`body` part). If the variation of the main template is described in the system, all other templates (`email.subject`, `email.from`, `push.title`) will be applied with the same variation. If there are multiple variations, they will be checked in some specified order (usually from more detail to less detail). If there are no variations, the base template will be used. In most cases there are no variations.

The following channels are available:

- `sms` - sending messages by SMS. There are no parts for this channel;
- `email` - sending messages by email. Parts for this channel:
  - `subject` - subject;
  - `body` - main content;
  - `from` - sender (optional);
- `push` - sending push notifications. Parts for this channel:

- title - subject;
- body - main content.

Example keys for `login_unknown_device` messages type:

- `message.notif.login_unknown_device.email.subject` - subject of the email message;
- `message.notif.login_unknown_device.email.body` - text of the email message;
- `message.notif.login_unknown_device.email.from` - sender of the email message;
- `message.notif.login_unknown_device.sms` - SMS text.

### Device and browser names

In Blitz Identity Provider you can customize the names of devices (operating systems) and browsers with exact version. To do this, you need to create lines in the `custom_messages` directory in the `messages` file whose names correspond to the following patterns:

- for browsers - `dics.browsers.<name>`. The following browsers are supported for substitution into `<name>`: Firefox, Opera, Chrome, Safari, IE, Edge, Yandex, Sputnik, unknown. The text of the string receives the browser version as a substitution string `{0}`.
- for devices (operating systems) - `dics.devices.<typ>.<os>.<ver>`. As `<typ>` you can specify: `kindle`, `mobile`, `tablet`, `iphone`, `windowsPhone`, `pc`, `ipad`, `playStation`, `unknown`. As `<os>` you can specify: `Android`, `iOS`, `WindowsPhone`, `Windows`, `macOS`, `Linux`, `ChromeOS`, `unknown`. If no private string is defined for `<os>` and `<ver>`, the more general string is taken. The operating system version is passed into the string text as a `{0}` substitution string.

Example lines:

```
dics.browsers.Firefox=Firefox Browser {0}
dics.browsers.Opera=Opera {0}
dics.browsers.Chrome=Google Chrome {0}
dics.browsers.Safari=Safari {0}
dics.browsers.IE=Internet Explorer
dics.browsers.Edge=Microsoft Edge {0}
dics.devices.mobile=Mobile device
dics.devices.mobile.Android=Android
dics.devices.mobile.Android.10=Android 10
dics.devices.mobile.Android.9=Android 9
dics.devices.tablet=Tablet
dics.devices.iphone=iPhone
dics.devices.iphone.iOS.14=iPhone (iOS {0})
dics.devices.pc.macOS=macOS {0}
dics.devices.pc.macOS.13=macOS Ventura {0}
dics.devices.pc.macOS.12=macOS Monterey {0}
dics.devices.pc.macOS.11=macOS Big Sur {0}
dics.devices.pc.macOS.10.15=macOS Catalina {0}
dics.devices.pc.macOS.10.14=macOS Mojave {0}
dics.devices.pc.macOS.10.13=macOS High Sierra {0}
dics.devices.pc.macOS.10.12=macOS Sierra {0}
dics.devices.pc.Windows.8=Windows 8
dics.devices.pc.Windows.10=Windows 10
dics.devices.pc.Windows.11=Windows 11
```

## Messages for different applications

It is possible to modify all text messages and templates in order to use specific texts/templates for different applications. For example, you can brand emails sent during registration on different websites connected to the same Blitz Identity Provider installation, or provide a link to download different resource rules.

To bind a set of templates to a specific application, follow the steps:

1. Create a text file copy that should be used only for this application. To do this, create a text file `messages.ru-special1` (`messages.en-special1`) in the `custom_messages/` directory for this application, in which `special1` is a sequence of 5-8 characters (both numbers and letters of the Latin alphabet are allowed).
2. Edit the `messages.ru-special1` (`messages.en-special1`) file to [add](#) (page 234) application-specific strings. All other strings will be taken from the default string database.
3. Edit the `blitz.conf` file as follows:
  - in the `blitz.prod.local.idp.apps` section of the file, find the application ID that should use the created template file;
  - add a parameter to the application settings in the `"lang-variant" : "special1"` format, in which `special1` is the character sequence used to label the template.

Example:

```
"demo-application" : {
  "domain" : "http://testdomain.ru",
  "lang-variant" : "special1",
  "name" : "test",
  "oauth" : {
    "autoConsent" : false,
    "clientSecret" : "1234567890",
    "defaultScopes" : [],
    "enabled" : true,
    "redirectUriPrefixes" : [
      "http://localhost"
    ]
  },
  "theme" : "default"
}
```

4. In the `blitz.prod.local.idp.lang -> lang-variant` setting, register all character sequences used to label various applications (`special1`, `special2`).

After that, a specially created message file will be used when logging into this application.

## Auxiliary application messages (pipes)

In Blitz Identity Provider, you can configure the messages of the helper application that issues the security key (Passkey, WebAuthn, FIDO2) at user login. You can configure different message texts depending on the user's devices (operating systems). To do this, create strings in the `custom_messages` directory in the `messages` file whose names correspond to the following patterns:

- `pipes.conf.webAuthn.addKey.<message-path>.<device-type>.<os>`;
- `login.outside.flow.error.internal.webAuthn.addKey.<device-type>.<os>`.

As `<message-path>` the string name is specified (see example below). The `<device-type>` specifies the device type: `mobile`, `tablet`, `iphone`, `pc`, `ipad`. As `<os>` you can specify: `Android`, `iOS`, `Windows`, `macOS`, `Linux`, `ChromeOS`. If no private string is defined for `<device-type>` and `<os>`, the more general string is taken.

Example lines:

```

pipes.conf.webAuthn.addKey.page.title.pc.macOS=Log in with Touch ID
pipes.conf.webAuthn.addKey.head.title.pc.macOS=Log in with Touch ID
pipes.conf.webAuthn.addKey.info.pc.macOS=Use Touch ID or MacOS password to log in_
↳to applications?
pipes.conf.webAuthn.addKey.finishInfo.pc.macOS=Log-in with Touch ID is configured_
↳for your account. Click Next
pipes.conf.webAuthn.addKey.name.pc.macOS=Touch ID on Mac
login.outside.flow.error.internal.webAuthn.addKey.pc.macOS=Error when configuring_
↳log-in with Touch ID

pipes.conf.webAuthn.addKey.page.title.pc.Windows=Log in with Windows Hello
pipes.conf.webAuthn.addKey.head.title.pc.Windows=Log in with Windows Hello
pipes.conf.webAuthn.addKey.info.pc.Windows=Use PIN, facial recognition, or a_
↳fingerprint to log in to applications?
pipes.conf.webAuthn.addKey.finishInfo.pc.Windows=Log-in with Windows Hello is_
↳configured for your account. Click Next
pipes.conf.webAuthn.addKey.name.pc.Windows=Windows Hello
login.outside.flow.error.internal.webAuthn.addKey.pc.Windows=Error when_
↳configuring log-in with Windows Hello

pipes.conf.webAuthn.addKey.page.title.iphone.iOS=Log in with Face ID
pipes.conf.webAuthn.addKey.head.title.iphone.iOS=Log in with Face ID
pipes.conf.webAuthn.addKey.info.iphone.iOS=Use Face ID or Touch ID on the phone to_
↳log in to applications?
pipes.conf.webAuthn.addKey.finishInfo.iphone.iOS=Log-in with Face ID is configured_
↳for your account. Click Next
pipes.conf.webAuthn.addKey.name.iphone.iOS=Face ID на iPhone
login.outside.flow.error.internal.webAuthn.addKey.iphone.iOS=Error when_
↳configuring log-in with Face ID

pipes.conf.webAuthn.addKey.page.title.ipad.iOS=Log in with Touch ID
pipes.conf.webAuthn.addKey.head.title.ipad.iOS=Log in with Touch ID
pipes.conf.webAuthn.addKey.info.ipad.iOS=Use Touch ID on iPad to log in to_
↳applications?
pipes.conf.webAuthn.addKey.finishInfo.ipad.iOS=Log-in with Touch ID is configured_
↳for your account. Click Next
pipes.conf.webAuthn.addKey.name.ipad.iOS=Touch ID on iPad
login.outside.flow.error.internal.webAuthn.addKey.ipad.iOS=Error when configuring_
↳log-in with Touch ID

pipes.conf.webAuthn.addKey.page.title.mobile.Android=Log in with facial_
↳recognition or fingerprint
pipes.conf.webAuthn.addKey.head.title.mobile.Android=Log in with facial_
↳recognition or fingerprint
pipes.conf.webAuthn.addKey.info.mobile.Android=Use facial recognition or_
↳fingerprint to log in to applications?
pipes.conf.webAuthn.addKey.finishInfo.mobile.Android=Log-in with facial_
↳recognition or fingerprint is configured. Click Next
pipes.conf.webAuthn.addKey.name.mobile.Android=Smart Lock on Android
login.outside.flow.error.internal.webAuthn.addKey.mobile.Android=Error when_
↳configuring log-in with facial recognition or fingerprint

pipes.conf.webAuthn.addKey.page.title=Log in with security key
pipes.conf.webAuthn.addKey.head.title=Log in with security key
pipes.conf.webAuthn.addKey.info=Use the FIDO2 security key to log in to_
↳applications?
pipes.conf.webAuthn.addKey.finishInfo=Log-in with security key is configured for_
↳your account. Click Next
pipes.conf.webAuthn.addKey.name=FIDO2

```

In Blitz Identity Provider, you can configure texts for an auxiliary application that shows a message to the user while login to the application. To do this, define in the `custom_messages` directory in the `messages` file

the strings for the customized `blitz.prod.local.idp.built-in-pipes.info` applications with their `{id}` of the helper application.

Example lines:

- `pipes.info.head.title.{id}`: tab name
- `pipes.info.page.title.{id}`: title of the auxiliary application
- `pipes.info.message.{id}`: message text
- `pipes.info.read.{id}`: button name (for auxiliary applications with the “news” type)
- `pipes.info.agree.{id}`: the name of the first button (for auxiliary applications with the “agreement” type)
- `pipes.info.disagree.{id}`: name of the second button (for auxiliary applications with the “agreement” type)

You can customize texts in Blitz Identity Provider for a helper application that asks the user to select a value from a list at user’s login and stores the result of the selection in an account attribute. To do this, define in the `custom_messages` directory in the `messages` file the strings for the configured `blitz.prod.local.idp.built-in-pipes.choice` applications with their `{id}` of the helper application.

Example lines:

- `pipes.choice.head.title.{id}`: tab name
- `pipes.choice.page.title.{id}`: title of the auxiliary application
- `pipes.choice.info.{id}`: text of the information under the title
- `pipes.choice.button.{id}.{choiceId}`: text on the selection button
- `pipes.choice.skip`: text on the skip button

You can customize texts in Blitz Identity Provider for an auxiliary application that asks the user to enter an attribute value at application login. To do this, define lines in the `custom_messages` directory in the `messages` file that correspond to the following pattern - `pipes.act.attr.<message-path>.common.<attr-name>`. The string name is specified as `<message-path>` (see below for an example). The attribute name is specified as `<attr-name>`.

Example strings (in case the `family_name` attribute is filled):

```
pipes.act.attr.page.title.common.family_name=Confirm your last name
pipes.act.attr.head.title.common.family_name=Confirm your last name
pipes.act.attr.info.confirm.common.family_name=Is this your last name?<br>If so,
↪click <b>Confirm</b>.
pipes.act.attr.info.enter.common.family_name=Your account doesn't contain the last
↪name.<br>Specify it and click <b>Confirm</b>.
pipes.act.attr.label.common.family_name=Last name
pipes.act.attr.msg.required.msg.common.surname=Enter your last name
```

### 2.5.5 Logos for external provider log-in buttons

In Blitz Identity Provider, you can change the logos displayed on the login buttons using external identity providers (social networks) on the login page and the external identity provider bind buttons in the User profile.

To customize, you must create lines in the `custom_messages` directory in the `messages` file whose names correspond to the following patterns:

- for the login page is `meth-logo.${type}.${name}`
- for User profile - `social-icon.${type}.${name}`



`${type}` specifies the type of external identity provider, `${name}` specifies the name of the identity provider. The values are taken from the [настройка](#) (page 109).

The string values specify the `<icon class>` names assigned to buttons.

Example lines:

```
social-icon.saml.demo-idp=saml-demo
meth-logo.saml.demo-idp=meth-saml-demo
```

## 2.6 Configuration file settings

### 2.6.1 Configuration file list

Blitz Identity Provider settings, except for `blitz-keeper` application, are located in the directory `/usr/share/identityblitz/blitz-config`.

List of subdirectories and files:

- `apps/` – [settings of connected applications](#) (page 277);
- `assets/` - user interface strings. See:
  - [Using and updating the plug-in](#) (page 73),
  - [External identity providers](#) (page 109),
  - [Login page](#) (page 221);
- `custom_messages/` - [user interface strings](#) (page 234);
- `devices/` - [auxiliary directories for handling HOTP and TOTP device loading](#) (page 93);
- `/dynamic/idstore/` – [custom procedures for customizing the logic of operations with data storages](#) (page 215);
- `flows/` - [login procedures](#) (page 191);
- `saml/` - [SAML settings](#) (page 167);
- `simple/` - [settings for connecting applications using Simple protocol](#) (page 157);
- `token_exchange/rules/*` - [access token exchange rules settings](#) (page 451);
- `blitz.conf` is the [main configuration file](#) (page 245);
- `boot.conf` - configuration file path settings;
- `console.conf` - [admin console settings](#) (page 279);
- `credentials` - [admin console administrator user profiles](#) (page 281);
- `play.conf` – application server settings. See:
  - installation of the admin console `blitz-console` in [General installation instructions](#) (page 14),
  - [Blitz Identity Provider domain](#) (page 262);
- `logback.xml` - event and error logging settings.

Most settings are set using the admin console. A number of settings require editing configuration files yourself. Such settings are described in the following subsections.

The `blitz-keeper` configuration file for the `blitz-keeper` application is located in `/etc/blitz-keeper`. The following configuration files are used:

- `blitz-keeper.conf` - [security gateway settings](#) (page 451);
- `blitz-keeper-log4j.xml` - event and error logging settings.

## 2.6.2 Settings in blitz.conf file

The main configuration file `blitz.conf` consists of the following configuration blocks with the following list of purposes:

- `blitz.prod.local.idp.apps` - settings of connected apps;
- `blitz.prod.local.idp.apps-source` – location of the connected application settings;
- `blitz.prod.local.idp.audit` - security event logging settings;
- `blitz.prod.local.idp.captcha` - settings for interaction with the CAPTCHA service;
- `blitz.prod.local.idp.events` - settings for sending events to the queue;
- `blitz.prod.local.idp.federation` - external identity provider settings;
- `blitz.prod.local.idp.flexible-flows` - login procedures settings;
- `blitz.prod.local.idp.id-attrs` - attribute settings;
- `blitz.prod.local.idp.id-stores` - attribute storage settings in the credential storage;
- `blitz.prod.local.idp.internal-store` - DBMS connection settings;
- `blitz.prod.local.idp.keystore` - key store access settings;
- `blitz.prod.local.idp.lang` – Blitz Identity Provider language settings;
- `blitz.prod.local.idp.license` is the Blitz Identity Provider license key;
- `blitz.prod.local.idp.logger` - logger settings;
- `blitz.prod.local.idp.login` - settings for authentication methods;
- `blitz.prod.local.idp.logout` - settings of the logout process;
- `blitz.prod.local.idp.messages` - message file settings;
- `blitz.prod.local.idp.messaging` - settings for invoking messaging services;
- `blitz.prod.local.idp.net` - network settings;
- `blitz.prod.local.idp.notifier` - event notification settings;
- `blitz.prod.local.idp.oauth` - scopes settings;
- `blitz.prod.local.idp.password-policy` - password policy settings;
- `blitz.prod.local.idp.play` - Blitz Identity Provider application server settings;
- `blitz.prod.local.idp.provisioning` - user registration and forgotten password recovery services settings;
- `blitz.prod.local.idp.realms` - Application ID encryption settings (privacy domains);
- `blitz.prod.local.idp.rights` – settings of the access rights;
- `blitz.prod.local.idp.saml` - SAML settings;
- `blitz.prod.local.idp.stores` - primary DBMS settings;
- `blitz.prod.local.idp.tasks` - settings of the task processing method;
- `blitz.prod.local.idp.user-profile` - user profile settings;
- `blitz.prod.local.idp.webAuthn` - security key settings;
- `home` - path to Blitz Identity Provider installation directory on the application server.

The following is a description of the settings that are inaccessible from the admin console, they can be configured by editing the `blitz.conf` configuration file.

## Logins and passwords

### Number of password verifications

You can set a limit on the number of simultaneous password authentications with the same user login in a period of time. The default setting is that Blitz Identity Provider allows no more than 3 authentications to the same login within 600 ms. To adjust the default settings, you must add the following block in the `blitz.conf` configuration file to the `blitz.prod.local.idp.login.methods.password` section:

```
"throughput": {
  "limit": 3,
  "window": 600
}
```

### Password change at login

If Blitz Identity Provider is connected to a writable account storage (the storage is not in read-only mode), then when a user logs in with an account from that storage, if the password policy requires the user to change their password, the user will be presented with a password change screen (asking them to enter their old and new password). Sometimes displaying the password change screen at login is not desirable. You can disable the screen by setting the following block of settings in the `blitz.conf` configuration file under `blitz.prod.local.idp.login.methods.password`:

```
"changePasswordMode": {
  "type": "except_for",
  "idStores":["ldap1", "ldap2"]
}
```

The `idStores` setting should list the identifiers of those account storages for which the user should not be prompted to change their password at login.

### System names of login and password fields

By default, Blitz Identity Provider names the login and password fields with the identifiers `login` and `password` on the login page. When implementing Blitz Identity Provider when migrating from an existing login system that used different field names, there may be a requirement that you need to keep the previously used field names in Blitz Identity Provider. This may be useful because some browsers that have saved user logins and passwords and use them for auto-substitution will be able to continue to auto-substitute the saved values even when the login system switches to using Blitz Identity Provider, as long as the domain of the login page and the name of the fields on the login page are preserved.

To set the required login and password field names, the following settings must be added to the `blitz.prod.local.idp.password` settings block:

- `loginInputName` - ID of the login input field on the login page;
- `passwordInputName` - ID of the password input field on the login page.

Example of configuration:

```
"password" : {
  ...
  "loginInputName" : "j_username",
  "passwordInputName" : "j_password",
  ...
}
```

## Attributes

### External attribute validator

If the capabilities provided by regular expression input value [conversion rules](#) (page 44) are not sufficient to implement the required business logic for validating the acceptability of an attribute value, the use of an external validator can be programmed and configured for the attribute.

To do this, you need to create a program with an external validator and build it into a JAR file.

The created JAR file should be copied to the servers with Blitz Identity Provider applications. The JAR file location address should be specified in the Java option `extensionsDir`.

Example:

```
export JAVA_OPTS="${JAVA_OPTS} -DextensionsDir=/usr/share/identityblitz/extensions"
```

In the `blitz.prod.local.idp.id-attrs.attrsMeta` attribute settings block, you must add `validators` block in the `source` block to the attribute description block for which you want to enable validation via an external validator:

- in the `className` setting, specify the address of the Java class that implements the `AttributeValidator` interface from the Blitz JDK;
- in the `conf` block, specify the settings to be passed to the validator.

Example of configuration:

```
"id-attrs" : {
  "attrsMeta" : [
    {
      {
        "class" : "verified-mobile",
        "format" : "string",
        "name" : "phone_number",
        "realmed" : false,
        "required" : false,
        "searchable" : true,
        "source" : {
          "validators" : [
            {
              "className" : "validator.MobileValidator",
              "conf" : {
                "conf1" : "value1"
              }
            }
          ],
          "type" : "idStore"
        },
        "unique" : false
      },
      ...
    ]
  }
}
```

## Attribute translator

You can associate a translator with an attribute that describes the attribute's conversion rules for reading from the LDAP directory and writing to the LDAP directory. In the attribute storage settings block in the `blitz.prod.local.idp.id-stores.list.mappingRules` section of the attribute matching settings, in the attribute description block for which you want to enable a translator, you must add the `translator` block with the `className` setting, in which you must specify the name of the Java class that implements the translation algorithm. The Java class must implement the implementation of the `LdapAttributeTranslator` interface from the Blitz JDK.

For some attributes from Active Directory, Blitz Identity Provider provides built-in Java classes:

- If you need to configure a translator for the `objectGUID` attribute from Active Directory so that this attribute is represented as a GUID string rather than in byte form, use the `com.identityblitz.idp.store.ldap.core.translator.ObjectGUIDTranslator` Java class.

Example of configuration:

```
"id-stores" : {
  "list" : [
    {
      ...
      "mappingRules" : [
        ...
        {
          "name" : "objectGUID",
          "storeAttr" : "objectGUID",
          "translator" : {
            "className" :
              "com.identityblitz.idp.store.ldap.core.translator.
↳ObjectGUIDTranslator"
          }
        },
        ...
      ]
    }
  ]
}
```

- If you need to configure a translator for the `objectSID` attribute to convert it to the string form, use the `com.identityblitz.idp.store.ldap.core.translator.ObjectSIDTranslator` Java class. The converted attribute is searchable, but the `LIKE` operation is not supported. It cannot be modified or set at creation.

Example of configuration:

```
"id-stores" : {
  "list" : [
    {
      ...
      "mappingRules" : [
        ...
        {
          "name": "objectSID",
          "storeAttr": "objectSID",
          "translator": {
            "className": "com.identityblitz.idp.store.ldap.core.
↳translator.ObjectSIDTranslator"
          }
        },
        ...
      ]
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```
    ]
}
```

Using a self-developed translator, it is necessary to create a program with an external translator and assemble it into a JAR file.

The created JAR file should be copied to the servers with Blitz Identity Provider applications. The JAR file location address should be specified in the Java option `extensionsDir`.

Example:

```
export JAVA_OPTS="${JAVA_OPTS} -DextensionsDir=/usr/share/identityblitz/extensions"
```

## CAPTCHA

To display the CAPTCHA service for logins and passwords you need to create a configuration file and load the required files (CSS and JS).

Changes in the configuration file must be done:

- in the `blitz.prod.local.idp.captcha` settings block. An example of a setting entry is shown below:

```
"captcha" : {
  "exampleCaptcha": {
    "operations": [
      {
        "call": {
          "headers": [
            "accept:application/json",
            "Authorization:Bearer ${cfg.bearerToken}"
          ],
          "method": "post",
          "url": "https://captcha.example.com/captcha/1.0.0/check?
↪uniqueFormHash=${ste.uniqueFormHash}&code=${ocp.code}&options[system]=${cfg.
↪system}&options[token]=${cfg.token}"
        },
        "check": {
          "errRegExp": {},
          "okRegExp": {
            "error": "0"
          }
        },
        "name": "check",
        "newState": {
          "uniqueFormHash": "${rsp.result.uniqueFormHash-}"
        }
      },
      {
        "call": {
          "headers": [
            "accept:application/json",
            "Authorization:Bearer ${cfg.bearerToken}"
          ],
          "method": "get",
          "url": "https://captcha.example.com/captcha/1.0.0/create?type=${
↪cfg.type}&options[system]=${cfg.system}&options[token]=${cfg.token}"
        },
        "name": "create",
```

(continues on next page)

(continued from previous page)

```

        "newState": {
            "uniqueFormHash": "${rsp.result.uniqueFormHash-}"
        }
    },
    {
        "call": {
            "headers": [
                "accept:application/json",
                "Authorization:Bearer ${cfg.bearerToken}"
            ],
            "method": "post",
            "url": "https://captcha.example.com/captcha/1.0.0/refresh?
↪uniqueFormHash=${ste.uniqueFormHash}&type=${cfg.type}&options[system]=${cfg.
↪system}&options[token]=${cfg.token}"
        },
        "name": "refresh"
    }
],
"plainParams": {
    "type": "arithmetic"
},
"secureParams": {
    "bearerToken": "<access_token>",
    "system": "<system_id>",
    "token": "<system_token>"
}
}
}

```

This block contains parameters for calling three methods of CAPTCHA service (create, check, refresh), as well as secret parameters - access token (`bearerToken`), system identifier (`system`), and system token (`token`).

- in the block of login settings `blitz.prod.local.idp.password`. Inside this block add `captcha` block and configure it according to the example:

```

"captcha" : {
    "enabled": true,
    "initJs": "require(['https://demo.reaxoft.ru/themes/default/assets/js/
↪passwordCaptcha.js', 'captcha-conf'], function(captcha, conf){ captcha(conf,
↪'https://demo.reaxoft.ru/themes/default/assets/css/passwordCaptcha.css'});});",
    "mode": {
        "type": "always_on"
    },
    "name": "exampleCaptcha"
}

```

In this block you should configure the following parameters:

- `enabled` - indicates whether CAPTCHA is enabled or not (true/false);
- `initJs` - contains links to the Javascript and CSS loaded on the login page and required to display/invoke CAPTCHA on login page;
- `mode` - CAPTCHA display mode, the following modes are provided:
  - `always_on` - CAPTCHA is always displayed;
  - `on_header` - CAPTCHA is displayed if the request has the header specified in the `name` parameter and the value specified in the `value` parameter.
  - `by_brute_force_protection` - A CAPTCHA is displayed if Blitz Identity Provider has detected password brute force on a specific account or mass password brute force on all accounts.

When using `by_brute_force_protection` mode, it is required to additionally create `bruteForceProtection` settings block with the following settings in `blitz.prod.local.idp.password` block:

- `disabled` - whether the protection is disabled or not (true/false);
- `captcha` - whether to use the CAPTCHA test when the protection is triggered (true/false);
- `delay` - login delay time in seconds (applies if CAPTCHA usage is disabled);
- `block` system in the `thresholds` setting - if system-level protection is required (protection against brute force on different logins). The settings are:
  - `minAttemptsToActivate` - minimum number of passed inputs to activate the protection mechanism based on the system statistics (100 inputs by default);
  - `timeWindowInMin` - time window for collecting statistics on the ratio of successful and unsuccessful inputs in minutes, must be even (100 minutes by default);
  - `failedAttemptsPercent`, the `turnOff` setting is the threshold for turning off automatic protection, in percent;
  - `failedAttemptsPercent`, the `turnOn` setting is the threshold for turning on automatic protection, in percent.
  - `forced` - enable forced protection for all (true/false).
- `system` block in the `thresholds` setting - if protection at the level of individual users is required (protection against password mining for a particular user). The settings are specified:
  - `tTlInSec` - the period for which the counter of unsuccessful logins by user is accumulated in seconds (default is 3600 seconds);
  - `failedAttempts`, the `turnOn` setting is the number of invalid logins per period after which protection will be enabled for the account.

Example of `bruteForceProtection` block settings (only user-level protection is enabled):

```
"bruteForceProtection" : {
  "delay" : 0,
  "captcha" : true,
  "disabled" : false,
  "thresholds" : {
    "user" : {
      "failedAttempts" : {
        "turnOn" : 5
      },
      "tTlInSec" : 3600
    }
  }
}
```

Example of `bruteForceProtection` settings (user-level and system-level protection enabled):

```
"bruteForceProtection" : {
  "disabled": false,
  "delay" : 0,
  "captcha" : true,
  "thresholds" : {
    "system" : {
      "minAttemptsToActivate": 1000,
      "timeWindowInMin": 180,
      "failedAttemptsPercent" : {
        "turnOff" : 20,
        "turnOn" : 30
      },
    },
    "forced" : false
  }
}
```

(continues on next page)



(continued from previous page)

```

    },
    "user" : {
      "ttlInSec": 3600,
      "failedAttempts" : {
        "turnOn" : 5
      }
    }
  }
}

```

If you use Google's [reCAPTCHA v3](https://developers.google.com/recaptcha/docs/v3)<sup>45</sup> service as a CAPTCHA, you must:

- set the following settings in `blitz.prod.local.idp.captcha`:

```

"captcha" : {
  "reCAPTCHA v3" : {
    "operations" : [
      {
        "call" : {
          "headers" : [],
          "method" : "post",
          "url" : "https://www.google.com/recaptcha/api/siteverify?secret=${cfg.
↪secret}&response=${ocp.response}"
        },
        "check" : {
          "errRegExp" : {},
          "okRegExp" : {
            "score" : "1\\.0|0\\. (5|6|7|8|9)",
            "success" : "true"
          }
        }
      },
      {
        "name" : "verify"
      }
    ],
    "plainParams" : {
      "sitekey" : "SITE_KEY"
    },
    "secureParams" : {
      "secret" : "SITE_SECRET"
    }
  }
}

```

Instead of `SITE_KEY` and `SITE_SECRET` you should fill in the values obtained when registering Google reCAPTCHA v3 on the site <https://g.co/recaptcha/v3>. You should also adjust the value in the `score` parameter - set the required threshold for successful passing of the check (in the example, the threshold is set not lower than 0.5).

- set the following settings in `blitz.prod.local.idp.password.captcha`:

```

"captcha" : {
  "mode" : {
    "_name" : "X-Captcha-Check",
    "_value" : "true",
    "_type" : "on_header",
    "type" : "always_on"
  },
  "enabled" : true,
  "initJs" : "require(['/blitz/assets/blitz-common/javascripts/recaptcha_v3.js',

```

(continues on next page)

<sup>45</sup> <https://developers.google.com/recaptcha/docs/v3>

(continued from previous page)

```
↔'captcha-conf'], function(captcha, conf){ captcha(conf);});",
  "name" : "reCAPTCHAav3"
}
```

To add a CAPTCHA to the confirmation page for linking a user account to an account from an external identity provider, you must set the following settings in `blitz.prod.local.idp.externalIdps.captcha`:

```
"captcha" : {
  "mode" : {
    "_name" : "X-Captcha-Check",
    "_value" : "true",
    "_type" : "on_header",
    "type" : "always_on"
  },
  "enabled" : true,
  "initJs" : "require(['/blitz/assets/blitz-common/javascripts/recaptcha_v3.js',
↔'captcha-conf'], function(captcha, conf){ captcha(conf);});",
  "name" : "reCAPTCHAav3"
}
```

To add a CAPTCHA to the user registration page, you must set the following settings in `blitz.prod.local.idp.provisioning.registration.captcha`:

```
"captcha" : {
  "mode" : {
    "_name" : "X-Captcha-Check",
    "_value" : "true",
    "_type" : "on_header",
    "type" : "always_on"
  },
  "enabled" : true,
  "initJs" : "require(['/blitz/assets/blitz-common/javascripts/recaptcha_v3.js',
↔'captcha-conf'], function(captcha, conf){ captcha(conf);});",
  "name" : "reCAPTCHAav3"
}
```

To add a CAPTCHA to the password recovery page, you must set the following settings in `blitz.prod.local.idp.provisioning.recovery.captcha`:

```
"captcha" : {
  "mode" : {
    "_name" : "X-Captcha-Check",
    "_value" : "true",
    "_type" : "on_header",
    "type" : "always_on"
  },
  "enabled" : true,
  "initJs" : "require(['/blitz/assets/blitz-common/javascripts/recaptcha_v3.js',
↔'captcha-conf'], function(captcha, conf){ captcha(conf);});",
  "name" : "reCAPTCHAav3"
}
```

## Queue server

### Sending events to queue server

The following events can be sent to the queue server:

- user registration (USER\_REGISTERED);
- password changed (USER\_PASSWORD\_SET);
- marker of session cancellations changed (USER\_CRID\_CHANGED);
- user attribute changes (USER\_ATTRIBUTE\_CHANGED);
- clearing user attributes (USER\_ATTRIBUTE\_REMOVED);
- user removed (USER\_REMOVED);
- external user account bound (FEDERATION\_POINT\_BOUND);
- external user account detached (FEDERATION\_POINT\_UNBOUND);
- revocation of the authorization (scopes) issued to the application (SCOPES\_REVOKED);
- group created (GROUP\_CREATED);
- attributes of group updated (GROUP\_UPDATED);
- group removed (GROUP\_REMOVED);
- group member added (GROUP\_MEMBER\_ADDED);
- group member removed (GROUP\_MEMBER\_REMOVED).

To send events to the queue you should create a block `blitz.prod.local.idp.events` with the following code (using the example of user registration and password change):

```
"events" : {
  "drivers" : {
    "rabbit_driver" : {
      "properties" : {},
      "server" : {
        "host" : "<RMQ_HOST>",
        "port" : 5672
      },
      "type" : "RMQ",
      "user" : {
        "password" : "<RMQ_PASS>",
        "username" : "<RMQ_USERNAME>"
      }
    }
  },
  "routes" : {
    "USER_PASSWORD_SET" : [
      "password_sync"
    ],
    "USER_REGISTERED" : [
      "registration"
    ]
  },
  "targets" : [
    {
      "discardList" : "PSWD_SYNC_DISCARD",
      "driver" : {
        "ext" : {
          "exchange_name" : "users",
          "routing_key" : "pwd_sync"
        }
      }
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```

        },
        "id" : "rabbit_driver"
    },
    "encCertificate" : "rmqkey",
    "name" : "password_sync",
    "redelivery" : 3
},
{
    "discardList" : "REG_DISCARD",
    "driver" : {
        "ext" : {
            "exchange_name" : "users",
            "routing_key" : "registration"
        },
        "id" : "rabbit_driver"
    },
    "encCertificate" : "rmqkey",
    "name" : "registration",
    "redelivery" : 3
}
]
}

```

Following settings should be configured:

- RMQ\_HOST - RabbitMQ queue server domain;
- RMQ\_USERNAME - user name for the queue server;
- RMQ\_PASS - password for the queue server.

In addition, to encrypt passwords sent to the queue (only for `USER_REGISTERED` and `USER_PASSWORD_SET` messages), the `encCertificate` parameter should specify the alias of the electronic signature key (in the standard `BlitzIdPKeystore.jks` key store) with which to encrypt passwords in messages.

### Queue server as a message broker

Blitz Identity Provider uses a built-in message broker to handle asynchronous tasks, using a database to track tasks.

If the intensity of requests to the Blitz Identity Provider is high, it may be appropriate to use the RabbitMQ queue server as a message broker. To do this, you need to make the following settings in the RabbitMQ console (usually, `http://hostname:15672/`):

- create a queue with the name `blitz-tasks` (in the “Queues” menu of the console);
- create an exchange named `blitz-tasks-exh` (in the “Exchanges” menu of the console) and configure binding on the `blitz-tasks` queue with a `routing_key` named `blitz-tasks`;
- create the `blitz` user (in the “Admin” menu of the console) and assign rights to the created queue to it.

After configuring RabbitMQ, adjust the settings in `blitz.conf` - in the `blitz.prod.local.idp.tasks` block set `broker-type` to `rmq` and set the connection settings to RabbitMQ in the `broker-rmq` block:

- set the name `blitz-tasks-exh` in the `exchange` parameter;
- set the `queue` parameter in the `executionRules` block and the `name` parameter in the `queues` block to `blitz-tasks`;
- set the user name (`blitz`) in the `username` parameter in the `user` block;
- set the user’s password in the `password` parameter in the `user` block - the password will be encrypted after Blitz Identity Provider is launched;

- specify the address and port of the connection to RabbitMQ in the `host` and `port` parameters of the `server` block;
- if necessary, adjust other parameters defining the size of the connection pool (`poolSize`), the number of channels (`channelSize`), the waiting time for a response from the queue server (`ackTimeout`);
- if necessary, adjust the task processing broker settings that determine the number of attempts (`maxAttempts`) to re-process tasks in case of an error, the time between attempts (`redeliveryDelayInSec`), the size of the processed message bundle (`dequeueBatchSize`), the queue check period (`dequeuePeriodInSec`), the number of handlers (`executorPoolSize`):

A configuration example is shown below:

```
"tasks" : {
  "broker-type" : "rmq",
  "broker-rmq" : {
    "consumer" : {
      "poolSize" : 2
    },
    "exchange" : "blitz-task-exh",
    "publisher" : {
      "ackTimeout" : 15,
      "channelsSize" : 8,
      "poolSize" : 2
    },
    "server" : {
      "host" : "RMQ_HOST",
      "port" : 5672
    },
    "user" : {
      "password" : "CHANGE_ME",
      "username" : "blitz"
    }
  },
  "executionRules" : [
    {
      "maxAttempts" : 2,
      "queue" : "blitz-tasks",
      "redeliveryDelayInSec" : 60
    }
  ],
  "queues" : [
    {
      "dequeueBatchSize" : 10,
      "dequeuePeriodInSec" : 30,
      "executorPoolSize" : 5,
      "name" : "blitz-tasks"
    }
  ]
}
```

## Stores and databases

### Storing objects in Couchbase

You can reassign the internal Blitz Identity Provider storages (`buckets`) in Couchbase Server DBMS used for data storage. For the following datasets, you can specify the need to use other storages (`buckets`) than the default ones.

To configure other storages (`buckets`) you need to add settings in `blitz.prod.local.idp.internal-store-cb` block:

- `buckets` - list of used storages (`buckets`), if they are different from default ones;
- `bucketsMapping` - overriding the default dataset locations to store in other storages.

An example of the configuration file is shown below. As a result, the `acl` dataset is placed in the `users` storage, and `clt` and `iat` are placed in `apps`. By default, all three datasets were written to the `oauth` store.

```
"internal-store-cb" : {
  ...
  "buckets" : {
    ["users", "oauth", "audit", "builtin_idstore", "ctxs"]
  },
  "bucketsMapping" : {
    "acl" : "users",
    "clt" : "apps",
    "iat" : "apps"
  },
  ...
}
```

### Reading the Couchbase Server cluster configuration

If one of the Couchbase Server cluster nodes is unavailable, users may experience errors when logging in to Blitz Identity Provider. In this case, you must calibrate the value of the global cluster configuration read interval. If the connection to a node is interrupted, the configuration will be recalculated on time so that Blitz Identity Provider will only access the working nodes. Do the following:

1. Open the `/usr/share/identityblitz/blitz-config/blitz.conf` configuration file.
2. In the `blitz.prod.local.idp.internal-store-cb.ioConf` section, set the value of the `configPollInterval` parameter in milliseconds.

```
"internal-store-cb" : {
  "ioConf" : {
    "configPollInterval" : 2500
  },
  ...
}
```

## Object storage time

You can set a limitation of database records retention period for audit data (by default, records are stored indefinitely). To do this, in the `blitz.prod.local.idp.internal-store-cb` block, add the `ttlMapping` setting specifying the `doc_type` of the record (`aud`) and the retention time in seconds.

Configuration example (audit retention time is limited to 90 days):

```
"internal-store-cb": {
  ...
  "ttlMapping": {
    "aud": 7776000
  },
  ...
}
```

You can configure a limitation on the retention period of records in the database for device data. To do this, add the settings in the `blitz.prod.local.idp.login` block:

- `uaActiveTtlInSec` - storage time of the record of the device (in seconds) with which the user's long-term session is associated or which the user marked as trusted at login. If the setting is not specified, the information about such a device is stored for one year from the last login from this device;
- `uaInactiveTtlInSec` - time for storing the record of other devices (in seconds). If the setting is not specified, the information about such a device is stored for 5 days.

Example of configuration:

```
"login": {
  ...
  "uaActiveTtlInSec": 2678400,
  "uaInactiveTtlInSec": 432000,
  ...
}
```

## Advanced PostgreSQL connection settings

For advanced management of the connection pool with PostgreSQL or another JDBC-enabled database, do the following:

1. Open the `/usr/share/identityblitz/blitz-config/blitz.conf` configuration file.
2. In the `blitz.prod.local.idp.internal-store-jdbc.pool` section, set the following options:

Option	By default	Description
testOnBorrow	true	Check a connection status before sending data. In the case of an error, the system removes the connection and selects the next one from the pool.
testOnCreate	false	Check a connection status after it has been created in the pool.
testOnReturn	false	Check a connection status after returning it to the pool.
testWhileIdle	false	Check an idle connection status. In the case of an error, the connection is removed from the pool.
timeBetweenEvictionRunsMillis	-1	Interval in milliseconds between check runs of an idle connection. The option affects testWhileIdle.
validationQuery	-	SQL query executed when checking a connection status from the pool, isValid() is used if the value is empty.

```
"internal-store-jdbc" : {
  "pool" : {
    "initial_size" : 5,
    "max_idle_conn" : 10,
    "max_total_conn" : 20,
    "max_wait_conn_ms" : 30000,
    "min_idle_conn" : 7
    "testOnBorrow" : true,
    "testOnCreate" : true,
    "testOnReturn" : true,
    "testWhileIdle" : true,
    "timeBetweenEvictionRunsMillis" : 30000,
    "validationQuery" : ""
  }
}
```

### 3. Restart the services.

```
sudo systemctl restart blitz-idp blitz-console blitz-recovery blitz-
↳registration
```

## Advanced LDAP connection settings

You can create settings for connection to attribute storages working via LDAP protocol in the admin console. At the same time, you can set LDAP connection pool settings through the admin console. Blitz Identity Provider will use the common connector pool settings to set up connections by each application that uses the connection to the storages. This can lead to a large number of LDAP connectors being created.

Using the `blitz.conf` configuration file, you can configure the parameters of the initial and maximum number of connectors in the context of different Blitz Identity Provider applications (for example, for the admin console you can set smaller values of connectors in the pool than for the authentication service). To do this, in the `blitz.prod.local.id-stores` block in the settings of the corresponding storage, along with the `initialConnections` and `maxConnections` settings, you can create settings of the form `initialConnections#BLITZ_APP` and `maxConnections#BLITZ_APP`, where `BLITZ_APP` is the name of the corresponding application (`blitz-console`, `blitz-idp`, `blitz-registration`, `blitz-recovery`).

An example of a setting where the admin console is set to a smaller connection pool size than for the other applications:



```

"id-stores" : {
  "list" : [
    {
      "type" : "LDAP",
      ...
      "initialConnections" : 10,
      "initialConnections#blitz-console" : 1,
      "maxConnections" : 20,
      "maxConnections#blitz-console" : 1
    }
  ]
}

```

When making LDAP queries to the Blitz Identity Provider attribute storage, Blitz Identity Provider takes an existing LDAP directory connection from the connection pool. After the query is completed, Blitz Identity Provider does not close the connection, but returns it back to the connection pool for reuse. This procedure for interacting with LDAP provides high performance, but requires keeping connections to the LDAP directory open for extended periods of time. Firewall or LDAP directory settings may prevent Blitz Identity Provider applications from keeping LDAP directory connections open for extended periods of time.

Blitz Identity Provider TCP connections to an LDAP directory can be closed without a negotiated connection termination, so that the LDAP directory will close the connection without Blitz Identity Provider being notified. When attempting to use such a connection from the pool, a long timeout may occur before Blitz Identity Provider considers the connection closed and removes it from the connection pool. To ensure that this situation does not affect users, Blitz Identity Provider provides an algorithm to periodically check the validity of open LDAP connections. The `healthCheckInterval` period (in milliseconds) is used to check the connection status, and the timeout time in the absence of LDAP directory response to the request is set by the `connectionTimeout` parameters (in milliseconds). The described mode of optimal interaction with the connection pool is enabled by default (setting `useSyncMode` to `false`). In case of unstable operation of connections with LDAP directory it is recommended to try to enable synchronous mode of interaction with the directory (set `useSyncMode` to `true`).

Examples of recommended settings are below:

```

"id-stores" : {
  "list" : [
    {
      "type" : "LDAP",
      ...
      "connectionTimeout" : 3000,
      "healthCheckInterval" : 300000,
      "useSyncMode" : false
    }
  ]
}

```

If several attribute storages are connected to Blitz Identity Provider at the same time, a situation may arise that when identifying and authenticating a user by login and password, several accounts, possibly belonging to different people, with matching logins may be detected in several storages. This situation should be avoided when implementing Blitz Identity Provider, and by default, if such a situation is detected, Blitz Identity Provider will issue a login error to the user indicating that there is an incorrect user account situation. However, in some cases there may be a situation when the implementation deliberately allows several accounts of different users in different storages to be found by one login. In this case, you can specify the `firstSucceeded` mode in the `authStrategy` setting in the `blitz.prod.local.idp.login` settings block. In this case all found accounts will be checked, and whichever of them matches the user's password first will be logged in with that account.

Example of configuration:

```

"login" : {
  "authStrategy" : {

```

(continues on next page)

(continued from previous page)

```

    "mode" : "firstSucceeded"
  },
  ...
}

```

## Geodatabase

You can connect to Blitz Identity Provider a database in [mmdb](#)<sup>46</sup> format with geodata. In this case, Blitz Identity Provider will record the country, region and city data corresponding to the IP address, as well as the latitude, longitude and radius of accuracy obtained from the geodatabase, in addition to storing the IP address, when logging security events and memorizing the user's devices and browsers.

The saved geodata will be shown to the administrator in the admin console. It is also possible to enable displaying geodata to the user in the "User profile" and include it in the texts of notifications sent by SMS or email.

To connect the database with geodata, you need to upload the mmdb file with the database to the servers with Blitz Identity Provider, and create a `blitz.prod.local.idp.geoIp` settings block with the following settings in the driver block:

- `type` - type of the base with geodata. Only `geoIp2-db` type is supported;
- `path` - path on the server to the file with geodatabase in `mmdb` format.

Example of configuration:

```

"geoIp": {
  "driver": {
    "type": "geoIp2-db",
    "path": "geoIp/GeoIP2-City.mmdb"
  }
}

```

## Several DBMSs usage

In Blitz Identity Provider you can configure simultaneous use of Couchbase Sever and PostgreSQL DBMS for storing different types of objects. To do this, specify the following settings in the `blitz.prod.local.idp.stores` settings block:

- `default_type` - the default DBMS used. Possible values: `cb` - Couchbase Server, `jdbc` - PostgreSQL or other relational DBMS with JDBC support;
- `list-of-types` - identifiers of Blitz Identity Provider object classes and used for placing corresponding DBMS objects (`cb` or `jdbc`). Only those object classes that are hosted in a DBMS other than the one specified in `default_type` should be included in the configuration. The following object classes are available:
  - `user-store` - user profile attributes;
  - `access-token-store` - security tokens;
  - `refresh-token-store` - refresh tokens;
  - `id-ext-store` - bindings of external identity providers;
  - `device-code-store` - Confirmation codes for OAuth 2.0 Device Authorization Grant;
  - `access-list-store` - user-granted permissions to applications;
  - `blitz-action-store` -contact confirmation codes (sms, email);

<sup>46</sup> <https://www.maxmind.com/en/geoip2-databases>

- oath-token-store - bindings of HOTP and TOTP one-time password generators;
  - oath-load-proc-store - history of downloaded descriptions of hardware HOTP and TOTP one-time password generators;
  - confirmation-request-store - requests for one-time passwords;
  - reg-context-store - user registration context;
  - reg-context-storef - user registration context;
  - id-store-maker - a built-in storage of user IDs;
  - rcv-ctx-store - user password recovery context;
  - db-client-store - dynamic clients;
  - db-client-storef - dynamic clients;
  - initial-token-store - IAT tokens;
  - user-agent-store - users' devices (browsers);
  - web-authn-key-store - security keys;
  - audit-store - security events;
  - task-store - asynchronous tasks.
- utils - list of utility modules required for the used DBMS type: `modules.CouchbaseModule` - for Couchbase Server, `modules.JDBCModule` - for PostgreSQL.

Example of two DBMS sharing settings:

```
"stores" : {
  "default-type" : "jdbc",
  "list-of-types" : {
    "access-token-store" : "cb",
    "refresh-token-store" : "cb",
    "user-agent-store" : "cb"
  },
  "utils" : [
    "modules.CouchbaseModule",
    "modules.JDBCModule"
  ]
}
```

### Blitz Identity Provider domain

You can change the Blitz Identity Provider domain by editing domain settings configuration file in the `blitz.prod.local.idp.net` settings block.

Example of configuration:

```
"net" : {
  "domain" : "demo.identityblitz.com"
}
```

If necessary, [change](#) (page 231) the domain setting in `blitz.prod.local.idp.lang` in the `portal-lang-cookie` block.

The example of configuration file excerpt:

```
"lang" : {
  ...
  "portal-lang-cookie" : {
    "domain" : "identityblitz.com",
    ...
  }
}
```

If necessary, you can change the path to applications (by default, applications are available using the `/blitz` path). You can edit the path in the `play.conf` configuration file. It is necessary to change the context parameter in the `play.http` block:

```
"http" : {
  "context" : "/blitz",
  ...
}
```

Change the Blitz Identity Provider domain and path in the `/blitz-config/saml/conf/relying-party.xml`, `/blitz-config/saml/metadata/idp-metadata.xml` files.

An example of changing settings in `relying-party.xml`:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<ns18:RelyingPartyGroup ...>
  <ns18:AnonymousRelyingParty
    provider="https://demo.identityblitz.com/blitz/saml"
    defaultSigningCredentialRef="IdPCredential"/>
  <ns18:DefaultRelyingParty
    provider="https://demo.identityblitz.com/blitz/saml"
    defaultSigningCredentialRef="IdPCredential">
    ...
  </ns18:DefaultRelyingParty>
  ...
</ns18:RelyingPartyGroup>
```

An example of changing settings in `idp-metadata.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<EntityDescriptor ... entityID="https://demo.identityblitz.com/blitz/saml">
  <IDPSSODescriptor ...>
    ...
    <ArtifactResolutionService
      Binding="urn:oasis:names:tc:SAML:1.0:bindings:SOAP-binding"
      Location="https://demo.identityblitz.com/blitz/saml/profile/SAML1/SOAP/"
      ↪ArtifactResolution"
      index="1"/>
    <ArtifactResolutionService
      Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
      Location="https://demo.identityblitz.com/blitz/saml/profile/SAML2/SOAP/"
      ↪ArtifactResolution"
      index="2"/>
    <SingleLogoutService
      Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
      Location="https://demo.identityblitz.com/blitz/saml/profile/SAML2/Redirect/"
      ↪SLO"
      ResponseLocation="https://demo.identityblitz.com/blitz/saml/profile/SAML2/"
      ↪Redirect/SLO"/>
    <SingleLogoutService
      Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Plain-Redirect"
      Location="https://demo.identityblitz.com/blitz/saml/profile/SAML2/Redirect/
```

(continues on next page)

(continued from previous page)

```

↔Plain/SLO"
    ResponseLocation=
        "https://demo.identityblitz.com/blitz/saml/profile/SAML2/Redirect/Plain/SLO
↔"/>
    <SingleLogoutService
        Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
        Location="https://demo.identityblitz.com/blitz/saml/profile/SAML2/SOAP/SLO" /
↔>
    ...
    <SingleSignOnService
        Binding="urn:mace:shibboleth:1.0:profiles:AuthnRequest"
        Location="https://demo.identityblitz.com/blitz/saml/profile/Shibboleth/SSO"/>
    <SingleSignOnService
        Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
        Location="https://demo.identityblitz.com/blitz/saml/profile/SAML2/POST/SSO"/>
    <SingleSignOnService
        Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST-SimpleSign"
        Location="https://demo.identityblitz.com/blitz/saml/profile/SAML2/POST-
↔SimpleSign/SSO"/>
    <SingleSignOnService
        Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
        Location="https://demo.identityblitz.com/blitz/saml/profile/SAML2/Redirect/
↔SSO"/>
    <SingleSignOnService
        Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Plain-Redirect"
        Location="https://demo.identityblitz.com/blitz/saml/profile/SAML2/Redirect/
↔Plain/SSO"/>
</IDPSSODescriptor>
<AttributeAuthorityDescriptor ...>
    ...
    <AttributeService
        Binding="urn:oasis:names:tc:SAML:1.0:bindings:SOAP-binding"
        Location="https://demo.identityblitz.com/blitz/saml/profile/SAML1/SOAP/
↔AttributeQuery"/>
    <AttributeService
        Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
        Location="https://demo.identityblitz.com/blitz/saml/profile/SAML2/SOAP/
↔AttributeQuery"/>
    ...
</AttributeAuthorityDescriptor>
</EntityDescriptor>

```

## Users

### Blocking inactive users

Blitz Identity Provider tracks the time of last user activity. It is possible to block user accounts that have been inactive for a long time. To activate this feature, run the `lockinactive.sh` script in cron. The script is located in the `/usr/share/identityblitz/blitz-console/bin` directory on the server with the `blitz-console` application. It is recommended to run the script once a day during minimal activity on the system. Before running the script it is necessary to edit it in a text editor - install:

- `inactive_period` - required period of inactivity (in days), after which the account should be blocked;
- `range_size` is the range of account coverage (in days), accounts whose last activity was between (current date - `inactive_period` - `range_size`) and (current date - `inactive_period`) will be blocked.

Blitz Identity Provider also allows you to automatically lock an account at the time of a login attempt if the account has been inactive for a long period of time. To enable this feature, add the `blitz.prod.local.idp.lock`

configuration block with the `inactivity` block having a `limit` setting in seconds that specifies the maximum allowed inactivity period after which the account will be locked out for inactivity when a login attempt is made. In the `checkInterval` setting, you can specify the minimum period in seconds, no more often than which the account will be checked for inactivity period when logging in.

Example of configuration:

```
"lock" : {
  "inactivity" : {
    "checkInterval" : 86400,
    "limit" : 31536000
  }
}
```

In the settings of the password recovery service, you can enable the mode that will allow unlocking an account locked due to inactivity in case of successful [recovery of a forgotten password](#) (page 134).

### Prohibit reuse of the remote user ID

Blitz Identity Provider keeps track of previously used user IDs so that they cannot be reused after a user account has been deleted for a specified period of time. To do this, add the following `remove` section to the `blitz.prod.local.idp.provisioning` block, specifying the number of days (`days`) during which the user ID cannot be used for re-registration:

```
"provisioning" : {
  ...
  "remove" : {
    "mode" : "keepRemovedId",
    "days" : 365
  }
}
```

### WebAuthn, Passkey, FIDO2, U2F provider certificates

See also:

- [Logging in via WebAuthn, Passkey, FIDO2](#) (page 65)
- [Login confirmation with WebAuthn, Passkey, FIDO2, U2F](#) (page 67)

Blitz Identity Provider allows you to remap the list of intermediate and root certificates of security key providers (WebAuthn, Passkey, FIDO2, U2F). To do this, in the `blitz.prod.local.idp.webAuthn.trustedStores` settings block, specify the settings containing the type (`type`), file path (`path`) and password (`password`) of access to the key container to be used to verify the signature of attestation objects generated during security key registration. The standard key container is automatically updated when new versions of Blitz Identity Provider are installed and contains current root and intermediate certificates of TPM modules, FIDO, as well as Apple and Google certificates required to verify the signature of attestation objects. If you want to restrict security keys to devices from specific vendors, you must remove unnecessary root and intermediate certificates from the key container.

Example of configuration:

```
"webAuthn" : {
  ...
  "trustedStores" : [
    {
      "password" : "*****",
      "path" : "webAuthn-trusted-ca.jks",
      "type" : "jKS"
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```

    },
    ],
    ...
}

```

## OIDC, SAML, and external identity providers

### OIDC Discovery service

Blitz Identity Provider automatically posts the [OIDC Discovery](#)<sup>47</sup> service according to the settings specified in Blitz Identity Provider. As part of the service, you can specify a documentation address for the OIDC service. To specify your own documentation address, you must specify the `serviceDocumentationUrl` setting in the `blitz.prod.local.idp.oauth` settings block with the value of the documentation reference address.

### Call addresses of external providers

When implementing Blitz Identity Provider, you may need to configure calls from Blitz Identity Provider servers to external identity provider handlers not directly, but through a proxy server. In this case, it is necessary to change the default addresses of handlers of external identity providers to the addresses registered on the proxy server. To correct the handler addresses, you need to change the values of the `authUri`, `tokenUri`, `dataUri` settings in the corresponding blocks of the external Identity Provider settings in `blitz.prod.local.idp.federation`.

Example of settings for logging in through an external Google provider:

```

"federation" : {
  "points" : {
    "google" : [
      {
        ...
        "authUri" : "https://accounts.google.com/o/oauth2/auth",
        "tokenUri" : "https://accounts.google.com/o/oauth2/token",
        "dataUri" : "https://www.googleapis.com/oauth2/v1/userinfo?alt=json",
        ...
      },
      ...
    ]
  }
}

```

### External SAML provider

Blitz Identity Provider allows you to configure login through an external identity provider running SAML 2.0.

To do this, in the `blitz.prod.local.idp.federations` configuration block, create an external provider `saml` with the following settings:

- `name` is the system name of the external identity provider;
- `humanReadableName` - description of the external identity provider;
- `clientId` is the service provider name (`EntityId`) assigned to Blitz Identity Provider when registering with the external SAML Identity Provider;

<sup>47</sup> <https://tools.ietf.org/html/rfc8414>

- `signAuthnReq` - specifies whether `|project|` should sign the SAML request sent to the external identity provider;
- `checkAssertionSign` - determines whether to check the signature of SAML assertions received from an external identity provider (for PROD environments, it is mandatory to include signature verification);
- `credentials` block with access settings to the key container used for signing requests to the SAML Identity Provider. It is configured optionally in case a separate key container is required for interaction with an external identity provider. If the setting is not specified, keys will be taken from the main keystore configured in the `blitz.prod.local.idp.keystore` block (in this case the name of the identity provider from the name setting will be used as `alias`).

The settings are made:

- `alias` is the name of the key in the container;
- `keystore` is a configuration block containing the container type (`type`), which can be JCEKS or BKS, as well as the container path (`path`) and the container password (`password`);
- `idpMetaPath` - path to the file where the external identity provider metadata (XML file with IDP metadata) is stored;
- `userMatching` configuration block - specifies the rules of account matching:
  - in the `type` setting is an indication that the basic (value `builder`) account binding setting is used;
  - in the `mapping` setting - rules for mapping accounts from the external SAML Identity Provider to accounts in Blitz Identity Provider;
  - in the `matchingRules` setting - rules for migrating SAML assertions from an external identity provider to account attributes in Blitz Identity Provider;
  - `requireLogInToBind` - feature *"Prompt the user to enter a login and password for binding if the account has not been identified"*;
  - `strictMatching` - a feature *"Require password input if account has been identified"*;
  - `uniqueMatching` - a feature *"Only one account by the configured matching rules should be found for binding"*.

Example of external identity provider settings:

```
"federation" : {
  "points" : {
    "saml" : [
      {
        "name" : "demo-idp",
        "humanReadableName" : "External SAML IDP",
        "clientId" : "login.company.com",
        "signAuthnReq" : true,
        "checkAssertionSign" : true,
        "_credentials" : {
          "alias" : "demo-idp",
          "keyStore" : {
            "password" : "*****",
            "path" : "demo-idp-key.jks",
            "type" : "JCEKS"
          }
        },
        "idpMetaPath" : "demo-idp-metadata.xml",
        "userMatching" : {
          "type" : "builder",
          "mapping" : [
            {
              "attr" : "urn:saml:mail",
              "master" : false,

```

(continues on next page)



(continued from previous page)

```

        "value" : "${email}"
      }
    ],
    "matchingRules" : [
      [
        {
          "attr" : "urn:saml:mail",
          "value" : "${email}"
        }
      ]
    ],
    "requireLogInToBind" : false,
    "strictMatching" : false,
    "uniqueMatching" : false
  }
}
],
...
}
}

```

After creating the external provider settings, it is necessary to include it in the list of available external identity providers. To do this, in the `blitz.prod.local.idp.login` settings block in the list of authentication methods (`methods`) in the list of external login providers `externalIdps` add the external provider `fedPoint` corresponding to the configured one.

Configuration example to enable an external identity provider with type `saml` and name `demo-idp`:

```

"login" : {
  ...
  "methods" : {
    ...
    "externalIdps" : {
      "idps" : [
        ...
        {
          "fedPoint" : "saml:demo-idp"
        },
        ...
      ],
      ...
    },
    ...
  },
  ...
}

```

[Customize the logo](#) (page 243) for the login button via the external login provider.

## Logging incomplete login attempts

In Blitz Identity Provider, all events are logged when the process that caused them has finished. This is normal for most events, because the processes are short-term.

Among all the events being recorded, there are some important events related to user login. If the login is successful, a security event is logged at the very end of the login process, indicating who logged in, where and when, what authentication methods were used, IP address, UserAgent and many other details.

The login process can be a complex, depending on the configurations made during implementation. It may not always be sufficient to only enter a username and password and an additional login confirmation is required, or during the login process the user will interact with auxiliary applications (pipes), for example, to update a contact, configure a passkey or answer a question about whether he/she trusts the device/browser. If a user stops logging in at any point during this process, it may not be completed, and as a result, an audit event will not be created for that incomplete process. Depending on at what point this happens, this could be a security issue. For example, if a user simply opened the login page and did not enter a username and password, then logging such an event in the security log is has no particular interest. But if the user entered the correct username and password, but got to a login confirmation screen that he didn't pass, then such a security event would be a good reason to record. Perhaps the malicious user was brute-forcing the password and was able to successfully pick it, but was unable to pass the second authentication factor. A security event would make this situation known if it were recorded and analyzed.

To activate event logging of unsuccessful (incomplete) logins it is necessary to add parameters in the `blitz.prod.local.idp.login` settings block:

- `postponeEnabled` - value `true` if the mechanism is enabled;
- `postponeTtl` - time in seconds after which a pending audit event is logged if the login has not been completed.

In case RabbitMQ is used to process tasks, you must make an additional queue named `<main queue name>-postpone` for the main task queue and set the following arguments for it:

```
x-dead-letter-exchange = <exchange in use>
x-dead-letter-routing-key = <main queue>
```

Also for the created queue, you need to configure `binding` to the `exchange` used.

## Transferring security events to file or Kafka

In Blitz Identity Provider, you can configure the logging of security events to one or more receivers. The configuration is set in the `blitz.prod.local.idp.audit` configuration block.

The following settings must be configured:

- `emitters` - defines the list of receivers of audit records. For each receiver the settings block is filled in:
  - `type` - receiver type. Possible values:
    - \* `audit-store` - the record is made in the DBMS;
    - \* `log` - the record is made to the logger with the name `AUDIT`.
  - `enabled` - optional setting - determines whether the receiver is enabled or not;
  - `include` - optional setting - lists the types of security events (see the table below), which are used to write to the receiver. If the setting is not specified, all security events are written;
  - `exclude` - optional setting - lists the types of security events (see table below) that should not be written to the receiver. If the setting is not specified, no events are excluded. If the setting is specified together with `include`, the list of events is first defined by the `include` setting, and then the events specified in `exclude` are excluded from it. It is recommended not to use both `include` and `exclude` settings together, but to use only one of them;

- `logger` - optional setting - specified only for the receiver with `log` type. Allows you to define the name of the logger. If the setting is not specified, the recording is made to the logger with the name `AUDIT`;
- `name` - optional setting - specified for receivers with types `log` and `kafka`. Specifies the name of the receiver, since multiple receivers can be configured for these receiver types. If the setting is not specified, `log` and `kafka` are used as receiver names;
- `bootstrapServers` - a mandatory setting for a receiver with type `kafka` - specifies a list of addresses for initial connection to the Kafka cluster;
- `topic` - mandatory setting for receiver with type `kafka` - the name of the Kafka topic to which the event should be sent;
- `securityProtocol` - optional setting for receiver with type `kafka` - in case of using SASL connection may not be specified. In case of connection via SSL, the value `SSL` must be specified in the setting. If TLS is not configured in Kafka, the value `SASL_PLAINTEXT` must be specified;
- `sasl` - optional configuration block for receiver with type `kafka` - specifies connection parameters when using SASL authentication to connect to Kafka:
  - \* `jaasConfig` is a connection string that can use the substitution parameters from `secureParams`;
  - \* `mechanism` is the value of `PLAIN`;
  - \* `secureParams` - block with parameters that will be encrypted in the configuration file when the server is started.

Block example:

```
"sasl": {
  "jaasConfig": "org.apache.kafka.common.security.plain.PlainLoginModule_
↪required username=\"alice\" password=\"${pswd}\"; ",
  "mechanism": "PLAIN",
  "secureParams": {
    "pswd": "Content will be encrypted at startup",
  }
}
```

- `ssl` - optional configuration block for receiver with type `kafka` - sets SSL parameters for connection to Kafka:
  - \* `enabledProtocols` - lines with the list of enabled protocols;
  - \* `keyStore` is a settings block with parameters for accessing Blitz Identity Provider key container. It contains `type`, `path`, `password` settings;
  - \* `trustedStore` - settings block with parameters of access to the container with trusted certificates. It contains `type`, `path`, `password` settings;
  - \* `keyPassword` - optional setting - the password to access the key.

Block example:

```
"securityProtocol" : "SSL",
"ssl" : {
  "enabledProtocols" : ["TLSv1.2,TLSv1.3"],
  "keyStore" : {
    "password" : "CHANGE-ME",
    "path" : "/etc/blitz-config/bip-d1app01-1.jks",
    "type" : "JKS"
  },
  "trustedStore" : {
    "password" : "CHANGE-ME",
```

(continues on next page)

(continued from previous page)

```

    "path" : "/etc/blitz-config/ca.jks",
    "type" : "JKS"
  },
  "keyPassword": "CHANGE-ME"
},

```

- tuning - optional settings block for a receiver with type kafka - specifies optional producer settings for interaction with Kafka. Parameter names must be specified with a dot as in the [Kafka](#)<sup>48</sup> documentation.

Block example:

```

"tuning": {
  "client.id": "BlitzKafka"
}

```

- emitAtLeastOneOf - optional setting - the list of receivers is specified, it is enough to record events in any of them for the operation to be considered successful;
- emitToAllOf - optional setting - specifies the list of receivers that must receive confirmation of successful event recording for the operation to be considered successful. If the emitAtLeastOneOf and emitToAllOf settings are not specified, then confirmation from all configured receivers is mandatory;
- emitTimeoutInSec - optional setting - defines the maximum response time from the receiver in response to event record requests. If the setting is not specified, the wait is 60 seconds.

Example of audit recording settings in log, DBMS and Kafka at the same time:

```

"audit": {
  "emitters": [
    {
      "type": "log",
      "name": "users-log",
      "enabled": true,
      "logger": "AUDIT",
      "exclude": ["admin_added", "admin_pswd_changed", "admin_removed",
→"admin_roles_changed",
      "config_changed"]
    },
    {
      "type": "log",
      "name": "admins-log",
      "enabled": true,
      "logger": "AUDITADMIN",
      "include": ["admin_added", "admin_pswd_changed", "admin_removed",
→"admin_roles_changed",
      "config_changed"]
    },
    {
      "type": "audit-store",
      "enabled": true
    },
    {
      "type" : "kafka",
      "enabled": true,
      "name" : "kafka",
      "include": ["login"],
      "bootstrapServers" : ["infra-kfk01:9443"],

```

(continues on next page)

<sup>48</sup> <https://kafka.apache.org/documentation/#producerconfigs>

(continued from previous page)

```

"topic" : "blitz_audit",
"securityProtocol" : "SSL",
"ssl" : {
  "enabledProtocols" : ["TLSv1.2,TLSv1.3"],
  "keyStore" : {
    "password" : "CHANGE-ME",
    "path" : "/etc/blitz-config/bip-app01.jks",
    "type" : "JKS"
  },
  "trustedStore" : {
    "password" : "CHANGE-ME",
    "path" : "/etc/blitz-config/ca.jks",
    "type" : "JKS"
  }
},
},
},
],
"emitAtLeastOneOf": ["users-log", "admins-log", "kafka"],
"emitToAllOf": ["audit-store"],
"emitTimeoutInSec": 30
}

```

When logging an audit to the logger, you can configure the logger using the `logback.xml` configuration file (see for more information <https://logback.qos.ch/documentation.html> `\_\_` ). Example of configuring the `AUDIT` logger in the `logback.xml` configuration file:

```

...
<appender name="AUDIT" class="ch.qos.logback.core.rolling.RollingFileAppender">
  <file>${dir.logs}/audit-${app.name}.log</file>
  <encoder>
    <pattern>%date - [%level] -[%file:%line] - %message%n%xException{20}</
↪pattern>
  </encoder>
  <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
    <fileNamePattern>${dir.logs}/archive/audit-${app.name}.%d{yyyy-MM-dd}.
↪log.gz</fileNamePattern>
    <maxHistory>90</maxHistory>
    <totalSizeCap>5GB</totalSizeCap>
  </rollingPolicy>
</appender>

<logger name="AUDIT" additivity="false">
  <appender-ref ref="AUDIT" />
</logger>
...

```

Example of a log entry:

```

2023-11-20 13:29:47,170 - [INFO] -[LoggerEventEmitterDriver.scala:37] - {"ip_st":
↪ "Tashkent", "ip": "213.230.116.179", "authnDone": "true", "process_id": "b80ca03e-4718-
↪ 44ff-9456-7d4255610eaa", "ip_ctr": "Ўзбекистан", "type": "login", "object_id": "BIP-
↪ 123456", "protocol": "oAuth", "subject_id": "BIP-123456", "auth_methods": "cls:password
↪ ", "session_id": "f8d85ba2-a26a-447f-b82e-944b9218abb8", "timestamp": 1700476187069,
↪ "ch_platform_version": "\"14.1.0\"", "ch_platform": "\"macOS\"", "ip_ct": "Tashkent",
↪ "id_store": "ldap01", "ip_lng": "69.2494", "ip_rad": "5", "ch_ua": "\"Google Chrome\";
↪ v=\"119\", \"Chromium\";v=\"119\", \"Not?A_Brand\";v=\"24\"", "user_agent":
↪ "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like
↪ Gecko) Chrome/119.0.0.0 Safari/537.36", "lp_id": "test-system", "id":
↪ "6056828858453673-600312119", "ip_lat": "41.3171", "client_auth_method": "redirectUri
↪ "}

```

Possible types of security events:

- `admin_added` - administrator added
- `admin_pswd_changed` - administrator password changed
- `admin_removed` - administrator removed
- `admin_roles_changed` - administrator roles changed
- `app_password_changed` - the password for the application is set
- `attribute_changed` - attribute added, changed or deleted
- `attribute_confirmed` - attribute confirmed
- `auth` - authentication performed (with OAuth 2.0 Resource Owner Password Credentials)
- `auth_failed` - authentication error
- `auth_req` - authentication request
- `authz_granted` - OAuth authorization issued
- `authz_rejected` - OAuth authorization denied
- `authz_revoked` - OAuth authorization revoked
- `bind_ext_account` - account is bound to an external account
- `config_changed` - changed configuration settings
- `duo_put` - Duo Mobile app is bound
- `duo_remove` - Duo Mobile application is unbound
- `grant_right` - assigning access rights
- `group_attr_changed` - user group's attribute has been changed or deleted
- `group_registered` - user group created
- `group_removed` - user group deleted
- `hotp_attached` - HOTP generator is bound
- `hotp_detached` - HOTP generator is unbound
- `internal_user_deleted` - account deleted
- `locked_methods_changed` - changed the list of blocked authentication methods
- `login` - login is performed
- `login_failed` - login error
- `login_stopped` - unsuccessful login
- `logout` - user is logged out
- `logout_req` - logout request
- `member_added` - the user is joined to the user group
- `member_removed` - user excluded from the user group
- `needed_password_change` - flag of password change necessity is set
- `recovery` - account access restored
- `recovery_fail` - access recovery failed
- `recovery_req` - access recovery is request
- `registration` - account is registered
- `registration_req` - registration is request

- `required_factor_changed` - changed user authentication mode
- `reset_user_password` - password set by administrator
- `reset_user_sessions` – log out devices (reset sessions)
- `revoke_right` - revoke access rights
- `send_email_code` - confirmation code sent to email
- `send_push_code` - confirmation code sent to Push
- `send_sms_code` - confirmation code sent by SMS
- `token_exchange_failed` - access token exchange denied
- `token_exchanged` - access token has been exchanged
- `token_granted` - access token issued
- `totp_attached` - TOTP generator is bound
- `totp_detached` - TOTP generator is unbound
- `unbind_ext_account` - account is unbound from external account
- `user_locked` - account locked
- `user_password_changed` - user password changed
- `user_sec_qsn_changed` – security question changed
- `user_sec_qsn_removed` – security question removed
- `user_unlocked` - account unlocked
- `web_authn_reg_key` - security key added
- `web_authn_revoke_key` - security key removed

The set of record attributes may vary depending on the type of security event and the specifics of the login process. Attribute assignments in the audit record are shown in the table below:

### Assigning attributes to an audit record

Attribute	Purpose and possible meanings
<code>id</code>	Security event log entry identifier
<code>type</code>	Security event type
<code>alt_pswd_cause</code>	The reason why the user was asked to change the password. Possible values: <ul style="list-style-type: none"> <li>• <code>password_expired</code> - password expired</li> <li>• <code>password_reset</code> - password should be changed at the first login</li> <li>• <code>password_policy_violated</code> - password does not comply with password policy</li> </ul>
<code>attr_name</code>	Installed, deleted or modified attribute name

continues on next page

Table 1 – continued from previous page

Attribute	Purpose and possible meanings
auth_methods	<p>Contains a list of authentication methods passed by the user. Possible values:</p> <ul style="list-style-type: none"> <li>• password - password authentication</li> <li>• spnego - login using an OS session</li> <li>• x.509 - login with electronic signature tool</li> <li>• qrCode - login by QR code</li> <li>• tls – login using HTTP headers of the proxy server</li> <li>• webAuthn - login or login confirmation with security keys</li> <li>• css - automatic login based on the results of user registration or password recovery</li> <li>• sms - one-time password by SMS</li> <li>• email - one-time email password</li> <li>• hotp - second factor of authentication with hardware key fob</li> <li>• totp is the second factor of authentication using the software TOTP confirmation code generator</li> <li>• externalIdps:&lt;type&gt;:&lt;name&gt; - login using an external identity provider (social networks etc.)</li> <li>• userApp - secondary authentication in the mobile application</li> <li>• outside_%NAME% - external logging in method named %NAME%</li> </ul> <p>The presence of the prefix <code>cls:</code> before the method means that the login was performed using a long-term session, and that the primary login previously used those login methods listed after <code>cls:</code></p>
auth_soft_id	Authenticator application (when logging in by QR code)
authnDone	Whether authentication was performed at this login
captcha_passed	An indication that a CAPTCHA was asked when logging in
client_auth_method	<p>A method for authenticating the application that invoked Blitz Identity Provider applications:</p> <ul style="list-style-type: none"> <li>• internal - for events invoked by internal Blitz Identity Provider applications</li> <li>• x.509 - for events triggered by SAML applications, provided that the SAML request came signed</li> <li>• Basic - for applications calling REST services that use Basic authorization</li> <li>• redirectUri - for applications that have identified themselves in the URL (e.g., specified their client_id in the URL parameter), but whose authentication has not been performed (it is not reliably known that this is actually calling Blitz Identity Provider for this particular application)</li> <li>• Bearer - using access_token for authentication by mobile app with dynamic client_id/client_secret</li> </ul>

continues on next page



Table 1 – continued from previous page

Attribute	Purpose and possible meanings
dcId	Dynamic <code>client_id</code>
device	device ID
deviceFingerprint	Device imprint
dTyp	Device type (for dynamic registration)
email	E-mail address
entry_point	The type of interface used to register the user: <ul style="list-style-type: none"> <li>• WEB - when registering from Blitz Identity Provider web application,</li> <li>• REST - when registering via Blitz Identity Provider REST services.</li> </ul>
error	Error (for unsuccessful events)
ext_account_id	External account ID
ext_account_name	Name of the external identity provider
ext_account_type	Type of External Identity Provider
failed_method	Indicates which authentication method the user was unable to pass
group_id	User group identifier
group_profile	User Group Usage Profile Identifier
id_store	User profile storage
ip	user IP address
ip_ctr	Country according to IP address
ip_st	Region according to IP address
ip_ct	City according to IP address
ip_lat	Latitude according to IP address
ip_lng	Longitude according to IP address
ip_rad	Surrounding by IP address
lp_id	The identifier of the application ( <code>EntityId</code> for SAML or <code>client_id</code> for OIDC) that invoked Blitz Identity Provider.
mobile	Mobile phone number
module	Identifier of the modified setting block
new_attr_value	New value of the installed or modified attribute
new_factor	New value of the attribute indicating the necessity to verify the second authentication factor
new_roles	Roles added to the administrator account
oauth_scopes	List of authorizations granted or revoked by the user
object_id	Identifier of the operation object (user for which the operation was performed)
old_attr_value	Previous value of the deleted or modified attribute
old_factor	Previous value of the attribute indicating the need to verify the second authentication factor
old_roles	Roles revoked from the administrator account
origin_app	Identifier of the application that initiated user registration or password recovery
process_id	Process ID
protocol	The protocol for the application to communicate with Blitz Identity Provider. Possible values: <ul style="list-style-type: none"> <li>• SAML - for SAML and WS-Federation</li> <li>• OAuth - for OpenID Connect and OAuth 2.0</li> <li>• simple - for proxy authentication-</li> <li>• internal - to log in to the User Profile (<code>_blitz_profile</code>)</li> </ul>

continues on next page

Table 1 – continued from previous page

Attribute	Purpose and possible meanings
pswd_changed	An indication that a password change was recommended
pswd_tmp_locked	An indication that there was a temporary lockdown
recovery_contact	The contact (email or cell phone number) specified during recovery
recovery_type	Password recovery type: email or mobile
right_name	Name of access right
roles	Administrator account roles
session_id	Unique identifier of the user session. Allows you to correlate all user events performed by the user within a common user session
subject_id	Identifier of the subject of the operation (the user who invoked the operation)
tags	Tag of assigned or revoked access right
timestamp	The date and time of the event. For example, 2022-11-04T17:49:58.384+0300
tried_old_pswd	An indication that a login attempt was made with a password from the saved password history (previous password)
used_login	Login used at sign in
user_agent	User device data (UserAgent)
wa_key_id	Security key identifier
wa_key_name	Security key name
withDelay	Entry delay was enabled

### Storing application settings in separate files

By default, the settings of all connected applications are stored inside the main configuration file `blitz.conf` in the `blitz.prod.local.idp.apps` section. If a large number of applications (hundreds) are to be connected to Blitz Identity Provider, then keeping application settings in separate configuration files can be more preferable. For this, you need to:

1. In the `/usr/share/identityblitz/blitz-config` settings directory, create a root directory that will store the application settings. By default, the `/usr/share/identityblitz/blitz-config/apps` directory will be used.
2. Inside the directory of application settings, create a directory for each application, observing the following rules:
  - the directory name must be created out of the application identifier (`appId`);
  - if the application identifier contains the `/` character, it must be substituted with `#` in the directory name;
  - if the application identifier contains the `:` character, it must be substituted with `%` in the directory name.

**Note:** For example, you need to create the `https%##example.com` directory for the application with the `https://example.com` identifier.

**Important:** Make sure to create directories for the service applications `_blitz_console`, `_blitz_idp`, `_blitz_reg`, `_blitz_recovery`, `_blitz_profile`.

3. Inside each application directory, create a file with the name `app.conf`, containing an application configuration from the original `blitz.conf`. The relevant section must be called `app` and not the `appId` value, as it was in `blitz.conf`. Later on, inside the application directory, a hidden `.snapshot` directory with backups of the old application configurations will also be created after each setting modification through the console or API.

The example of the `app.conf` configuration file:

```
#####
->#####
# version: 822
# modified: 2023-08-20 21:17:27 MSK
# author: admin
# ip: 127.0.0.1
# user agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.
->36 ...
#####
->#####
{
  "app": {
    "domain": "https://company.com",
    "name": "test app",
    "oauth": {
      ...
    },
    ...
  }
}
```

4. After migrating all existing application settings from `blitz.conf` to separate configuration files, set the application setting reading mode in the `blitz.prod.local.idp.apps-source` section of `blitz.conf`:

```
"apps-source": {
  "type": "filesystem",
  "dir": "apps"
}
```

5. Restart Blitz Identity Provider applications and try to sign in to the applications. If everything is alright, you can remove the application settings from the original `blitz.prod.local.idp.apps` block.

### SSO session duration

The duration of a user's SSO session may be affected by the `blc` cookie validity period on the Blitz Identity Provider side. By default, the `blc` cookie validity period is 10800 seconds. If the [maximum session duration](#) (page 60) exceeds this value, a user may be asked to log in again as soon as the cookie expires, even with an active SSO session.

To adjust the cookie validity period, add the `lstateTtlInSec` parameter to the `blitz.prod.local.idp.login` block of the `blitz.conf` configuration file with a value equal to or greater than the maximum session duration.

```
"lstateTtlInSec" : 20200
```

### 2.6.3 Admin console settings

The admin console is configured using the `console.conf` and `credentials` files. The following subsections describe the possible settings.

#### Logging in to admin console via SSO

You can configure Blitz Identity Provider admin console to log in via the OIDC Identity Provider. The provider can be the current installation of Blitz Identity Provider, a custom installation of Blitz Identity Provider, or even an external software if it is compatible with OIDC.

The following admin console login modes are supported:

- *standard mode* (page 41) by the account login/password created in the section Administrators;
- login via SSO;
- the hybrid mode of logging in when the administrator can log in both by the login/password in the standard mode and using SSO.

If you are using the SSO mode, you don't have to create administrator accounts in the section Administrators.

To configure the admin console login mode via SSO, do the following:

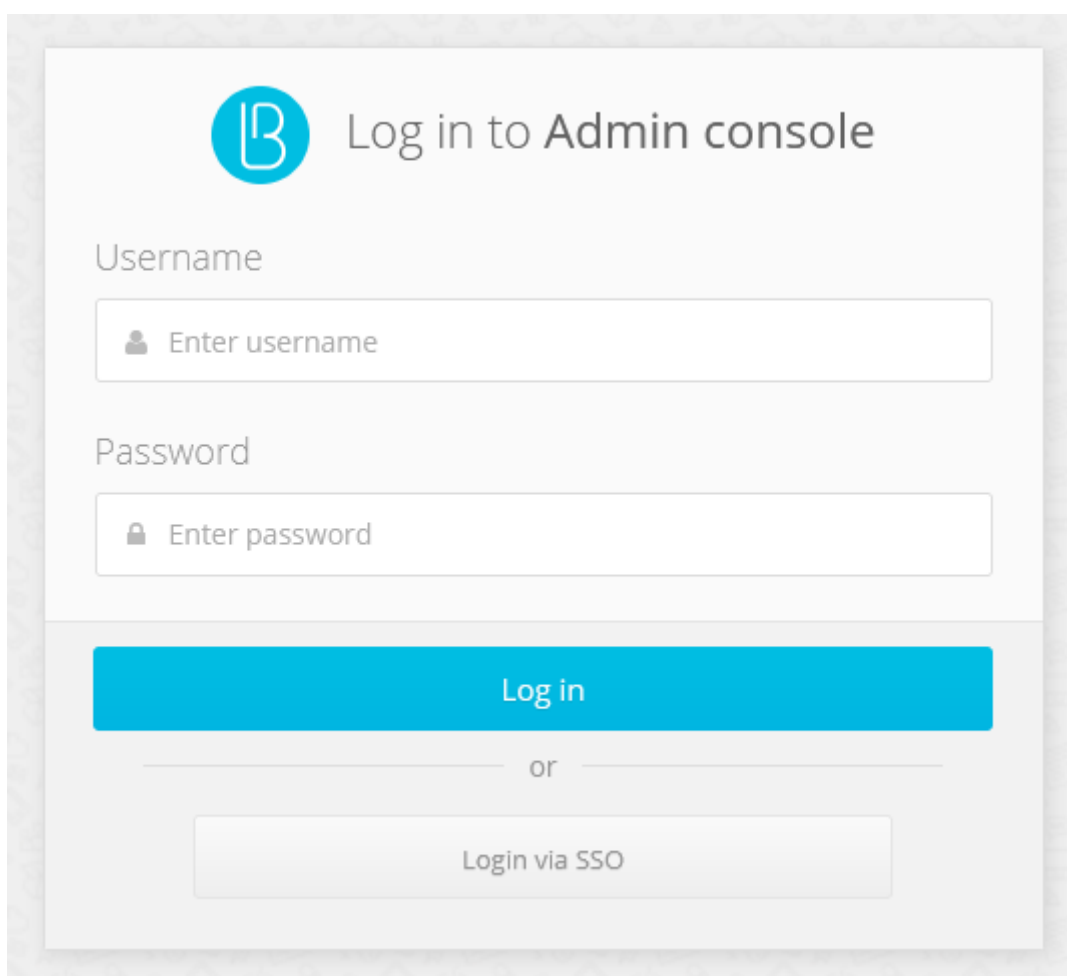
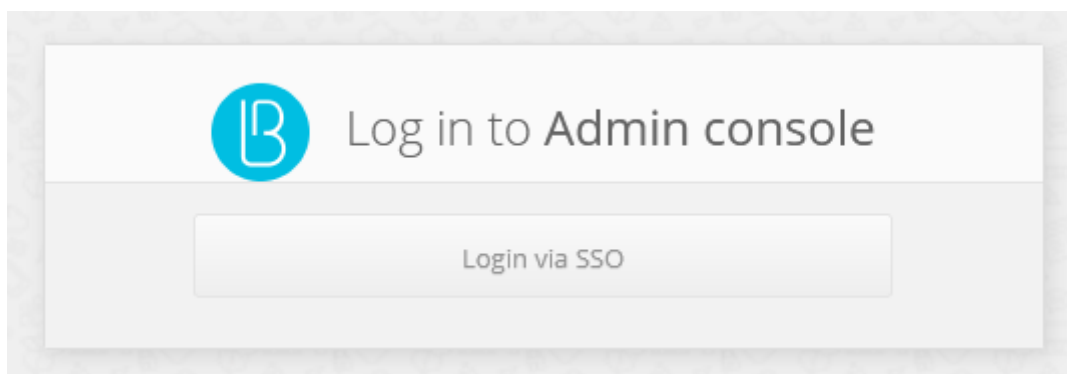
- In the SSO identity provider settings, that will be used to log in to the admin console, register the admin console as an application. In the allowed return prefixes (`redirect_uri`) specify the Blitz Identity Provider installation domain. As a result of the registration, you will get the `client_id` and `client_secret` parameters for the console;
- in the `console.conf` configuration file, create the `login` settings block with the following content:

```
{
  "login" : {
    "fp" : {
      "authUri" : "https://idp-host.com/blitz/oauth/ae",
      "clientId" : "blitz-console",
      "clientSecret" : "client_secret_value",
      "logoutUrl" : "https://idp host.com/blitz/login/logout?post_logout_redirect_
↩uri=https://idp host.com/blitz/console",
      "scopes" : [
        "openid"
      ],
      "subjectClaim" : "sub",
      "roleClaim" : "roles",
      "tokenUri" : "https://idp-host.com/blitz/oauth/te"
    },
    "mode" : "sso"
  }
}
```

You must specify parameters:

- In the `authUri` and `tokenUri` parameters, you must specify the addresses of the Authorization Endpoint and Token Endpoint handlers of the external identity provider.
- In the `clientId` and `clientSecret` parameters, specify the `client_id` and `client_secret` values, assigned to the application registered in the external identity provider application that corresponds to the admin console parameters.
- In the `logoutUrl` parameter, specify the link to which the user should be redirected user when logging out of the admin console, so that a single logout via an external identity provider.
- In the `scopes` parameter, prescribe the list of scopes which must be requested (at least `openid` scopes are needed).

- In `subjectClaim` specify the name of the attribute from the identity token (`id_token`) that is used as the account identifier. Using this identifier the administrator's login will be performed in `sso` login mode.
- Specify in `roleClaim` the attribute name from the identification token (`id_token`), in which the role (string) or the role list (array of strings) of the administrator is passed. Using these identifiers the administrator's login will be performed in `sso` login mode.
- In the `mode` parameter, you must specify the required login page mode: `sso` - login only using an external identity provider (see the figure below); `internal` - login only using the login and password from the administration console settings. If the parameter is not specified, both options are available at the user's choice. It is not required to create administrator accounts in the Administrators menu prior to logging in via SSO Logging in via SSO.



To avoid showing an intermediate login screen where the user clicks the Logging in via SSO button, you can in-

voke the admin console using a link of the following form: `https://hostname:port/blitz/console?mode=SSO`.

### Session limit

Security policy may require that a user or administrator cannot be logged in from multiple devices at the same time. To fulfill this security policy for administrator access to the admin console, the `session` block must be added to the `console.conf` configuration file:

```
"session" : {
  "mode" : "exclusive",
  "check-interval" : 10
}
```

With this setting, if an administrator login is detected with an account that has already logged in, the previous login will display the login page in the admin console for any action. The `check-interval` setting (specified in seconds) specifies the period of time in seconds how quickly the previous session will be logged out when a new session appears.

If the security policy also requires to prevent multiple sessions for normal users, this mode can be selectively enabled for certain users when logging in to certain applications. This can be done by configuring the [login procedure](#) (page 193).

Additionally, in the web application User profile it is necessary to enable the setting according to which an early log out from the web application will take place in case the user account is blocked or the policy prohibiting multiple user logins has been violated. In the `blitz.conf` configuration file, in the `blitz.prod.local.idp.user-profile` settings block, add the `check-session-interval` setting, which specifies the period of session activity check by the web application:

```
"user-profile" : {
  "check-session-interval" : 10,
  ...
}
```

### Roles and permissions for the console

The standard administrator roles are described in the [previous sections](#) (page 40). In the `credentials` configuration file you can create additional administrator roles or correct access rights in existing roles. To do this, in the `roles` block, adjust the composition of access rights (`privileges`) corresponding to the role (name).

Example of configuration:

```
"roles" : [
  {
    "name" : "new-role",
    "privileges" : ["w_app", "w_system", "w_ui", "w_user", "w_admin", "r_audit"]
  }
]
```

If new roles are created, [text strings with role names](#) (page 234) must also be defined for them. Example of a text string for a new role `new_role`:

```
page.admins.role.new-role=new role name
```

The list of available access rights for filling the `privileges` setting is given in the table below.

Blitz Identity Provider admin console access rights

Access rights	Available sections of the Admin console
w_app	Applications
w_sys-tem	Data sources, Authentication, Authentication flows, Identity providers, SAML, OAuth 2.0, Devices, Messages
w_ui	Self-services, Login page themes
w_admin	Administrators, Events
w_user	User search, Group search, Access rights
r_user	User search (read-only), Group search (read-only), Access rights (read-only)
r_audit	Events (read-only)

### Changing console admin password

To change the console administrator password, do the following:

1. Open the `/usr/share/identityblitz/blitz-config/console.conf` file.
2. Specify a new password in the `pswdHash` parameter in plaintext without encryption. The system will encrypt it after changes are applied.

```
"users" : [
  {
    "pswdHash" : "new$password",
    "roles" : [
      "root"
    ],
    "username" : "admin"
  }
]
```

3. Restart the `blitz-console` service.

```
sudo systemctl restart blitz-console
```

## 2.6.4 Configuring Token Exchange

Blitz Identity Provider supports the [OAuth 2.0 Token Exchange](#)<sup>49</sup> technology. A typical use case of this technology in Blitz Identity Provider is the interaction of the *Blitz Keeper* (page 451) security gateway with the authorization service to gain access to protected services.

To configure Token Exchange, follow the sequence described below.

### Step 1. Create service access rules

Token Exchange rules to access protected services are created in the `/usr/share/identityblitz/blitz-config/token_exchange/rules/` directory. Each rule is created as a separate text file without extension.

Example of a file with an access rule (the `specialize` type):

```
{
  "name": "rule-name",
  "type": "specialize",
  "desc": "",
```

(continues on next page)

<sup>49</sup> <https://tools.ietf.org/html/rfc8693>

(continued from previous page)

```

"subjectTokenCond": {
  "clientRights": [],
  "userRights": [],
  "scopes": ["openid"],
  "userClaims": {},
  "userGroups": []
},
"issue": {
  "ttlInSec": 3600,
  "allowedScopes": ["openid", "profile"],
  "allowedClaims": ["sub", "global_role", "org_id", "rights"],
  "addingScopes": [],
  "addingClaims": []
}
}

```

Example of a file with an access rule (the impersonate type):

```

{
  "name": "rule-name",
  "type": "impersonate",
  "desc": "",
  "subjectTokenCond": {
    "clientRights": [],
    "userRights": [],
    "scopes": ["openid"],
    "userClaims": {},
    "userGroups": []
  },
  "authClientCond": {
    "requiredRights": [
      {
        "rights": ["right1"],
        "target": {
          "type": "its",
          "name": "app1"
        }
      }
    ]
  },
  "issue": {
    "ttlInSec": 3600,
    "allowedScopes": ["openid", "profile"],
    "allowedClaims": ["sub", "global_role", "org_id", "rights"],
    "addingScopes": [],
    "addingClaims": []
  }
}

```

Following attributes of the access rule must be set:

- name - name of the rule, which must match the name of the file with the access rule;
- type – type of the rule. The following rule types are supported:
  - specialize – according to this rule, an application requests the exchange of an access token issued to the same application. The exchange is performed for the purpose of specializing the access token: replacing permissions (scope), attributes (claims), the list of recipients (audience, aud), or the token format (jwt or opaque);
  - impersonate – according to this rule, an application requests the exchange of an access token issued to another application. The exchange is carried out provided that in the access token being exchanged, the requesting application is in the list of recipients (audience, aud). Such an exchange is



used when application A initially received an access token, prepared it for transmission to application B (via an exchange using the `specialize` rule type), passed it on to application B, so that application B issued its own token based on the received one ( through an exchange using the `impersonate` rule type).

- `desc` - description of the rule. You can enter any text information;
- `subjectTokenCond` - conditions of rule fulfillment. If all the conditions specified in the rule are met, the rule is considered to be executed. If at least one of the conditions in the rule is not fulfilled, the whole rule is considered unfulfilled. The conditions of rule fulfillment can be as follows:
  - `clientRights` - check if the application has the specified access rights;

Example of the rule:

```
"clientRights": [
{
  "rights": ["right1"],
  "target": {
    "type": "its",
    "name": "app1"
  }
}
]
```

In this example, the calling application checks whether the calling application has the `right1` access right with respect to another application (`app1`). The `its` parameter in the `target` setting specifies the type of object against which the access right is checked. Possible values: `its` - right to an application; `grps` - right to an access group; no `type` - right to a user account.

- `userRights` - check if the user has the specified access rights.

Example 1 of a rule:

```
"userRights": [
{
  "rights": ["right2"],
  "target": {
    "type": "grps",
    "name": "org1",
    "ext": "orgs"
  }
}
]
```

In the example, it checks whether the user has the `right2` access right with respect to the user group (`org1`). In case of the access group object type, an additional parameter `ext` is specified to define the access group profile (see [Enabling the display of groups in `blitz.conf`](#) (page 149)).

Example 2 of a rule:

```
"userRights": [
{
  "rights": ["security_administrator"],
  "target": {
    "type": "grps",
    "name": "${org_id}",
    "ext": "orgs"
  }
}
]
```

This rule checks whether the user has the `security_administrator` access right with respect to the user group from the `orgs` profile, which has an identifier that matches the value of the `org_id` attribute from the original access token. In contrast to Example 1, this example illustrates the possibility to specify not a specific object value as the name of the access right object, but to refer to the object based on the values from the submitted access token (`$org_id`).

Example 3 of a rule:

```
"userRights": [
{
  "rights": ["right3"],
  "target": {
    "type": "its",
    "name": "app1"
  }
}
]
```

The above example checks if the user has the `right3` access rights with respect to application `app1`.

- `scopes` - checks the presence of the required permissions in the access token (see [General OAuth 2.0 settings](#) (page 175));

Example of the rule:

```
"scopes": ["scope1"]
```

This example checks if the original access token scopes with the name `scope1`.

- `userClaims` - checks that the attributes of the user account have specified values.

Example of the rule:

```
"userClaims": {"role": "FIN"}
```

This example checks if a user has the `role` attribute in the account with the `FIN` value filled in. Only attributes with `String` type can be used.

- `userGroups` - checks if the user account is part of the specified access groups.

Example of the rule:

```
"userGroups": [
{
  "name": "admin",
  "profile": "roles"
}
]
```

This example checks that the user is in the `admin` access group with `roles` profile.

- `authClientCond` – conditions for replacing `client_id`. These conditions are only checked for the `impersonate` rules. The rule verifies that the new application has the permissions to exchange the access token. The `requiredRights` condition is supported.

Example of the rule:

```
"requiredRights": [
{
  "rights": ["right1"],
  "target": {
    "type": "its",
    "name": "app1"
  }
}
]
```

(continues on next page)

(continued from previous page)

```

    }
  }
]

```

In this example, the calling application checks whether the calling application has the `right1` access right with respect to another application (`app1`). The `its` parameter in the `target` setting specifies the type of object against which the access right is checked. Possible values: `its` - right to an application; `grps` - right to an access group; no `type` - right to a user account.

- `issue` - rules for issuing a new access token, applied in case the rule was successfully executed. The rules for issuing a new access token consist of:
  - `ttlInSec` - the lifetime (in seconds) of the issued access token;
  - `allowedScopes` - the scopes that allowed in the issued access token;
  - `allowedClaims` - user attributes that allowed in the issued access token;
  - `addingScopes` - the scopes added to the access token;
  - `addingClaims` - user attributes added to the access token.

## Step 2. Configuring access token exchange

To define how access tokens will be exchanged over Token Exchange, specifically for which protected services which access rules should be applied, add the `blitz.prod.local.idp.token-exchange` configuration block to the `blitz.conf` configuration file as follows:

```

"token-exchange" : {
  "resources" : [
    {
      "uri" : "http://secured_service_host/api/service1",
      "methods" : ["GET", "POST"],
      "rules" : [
        "rule1",
        "rule2"
      ]
    },
    {
      "audience" : "secured-api",
      "rules" : [
        "rule3"
      ]
    },
    ...
  ]
}

```

In the `resources` block you need to fill in the settings for each service:

- `rules` – list the names of service access rules. Each rule corresponds to its own [configuration file](#) (page 282). Access to the service is allowed if at least one of the rules from this list is executed. If all the listed rules are not met, then access to the service will be denied;
- `uri` – optional parameter, can specify the address of the protected service. In specifying the service address it is allowed to use an asterisk (\*) to skip one component of the address path and a double asterisk (\*\*) to skip the rest of the service address path;
- `methods` - optional parameter, specifies the list of HTTP methods of the invoked service;

- `audience` – optional parameter, can specify the application name. This value will be included in the issued new access token in the `aud` attribute. It is mandatory to specify one of the parameters `uri` or `audience`.

## 2.7 Security, maintenance, and troubleshooting

### 2.7.1 Viewing security events

The Events section of the admin console is used to perform security auditing and to view security events logged in Blitz Identity Provider log. Here you can filter security events by various criteria:

- by user (specifying the user ID is required);
- by time period;
- by application name;
- by groups of events;
- by IP-address;
- by interaction protocols.

After filters are configured and applied, you can view detailed information about the various security events.

### 2.7.2 Application performance monitoring

#### Standard monitoring service

To monitor the availability of Blitz Identity Provider applications, invoke the `/blitz/metrics` service via HTTP GET. It is recommended that the service be available on each application server via HTTP when invoked from the monitoring servers located in the internal network and unavailable from external networks and user workstations.

If an application is available, the `/blitz/metrics` service will return its detailed performance metrics in the [Prometheus](https://prometheus.io/)<sup>50</sup> format.

<sup>50</sup> <https://prometheus.io/>

## Example of the service response

```

# HELP blitz_idp_uptime_seconds Uptime
# TYPE blitz_idp_uptime_seconds gauge
blitz_idp_uptime_seconds{blitz_host="papp01.loc",} 63859.0
# HELP blitz_idp_licence_exp_seconds Licence expiration
# TYPE blitz_idp_licence_exp_seconds gauge
blitz_idp_licence_exp_seconds{blitz_host="papp01.loc",} 9.223372036854776E18
# HELP blitz_idp_config_mtime Last time, a file was changed
# TYPE blitz_idp_config_mtime gauge
# HELP blitz_idp_datasource_latency Latency of an datasource operation
# TYPE blitz_idp_datasource_latency histogram
blitz_idp_datasource_latency_bucket{blitz_host="papp01.loc",ds_type="ldap",ds_name=
↪"389-ds",op_type="read",le="0.005",} 13.0
...
blitz_idp_datasource_latency_bucket{blitz_host="papp01.loc",ds_type="ldap",ds_name=
↪"389-ds",op_type="read",le="+Inf",} 29.0
blitz_idp_datasource_latency_count{blitz_host="papp01.loc",ds_type="ldap",ds_name=
↪"389-ds",op_type="read",} 29.0
blitz_idp_datasource_latency_sum{blitz_host="papp01.loc",ds_type="ldap",ds_name=
↪"389-ds",op_type="read",} 0.31127871899999999
# HELP blitz_idp_mq_connections Amount connections to datasource
# TYPE blitz_idp_mq_connections gauge
blitz_idp_mq_connections{blitz_host="papp01.loc",mq_type="rmq",mq_server="pmq01.
↪loc_5672",} 1.0
# HELP blitz_idp_mq_latency Latency of an mq operation
# TYPE blitz_idp_mq_latency histogram
blitz_idp_mq_latency_bucket{blitz_host="papp01.loc",mq_type="rmq",mq_server="pmq01.
↪loc_5672",broker="blitz.events.direct",op_type="write",le="0.005",} 1.0
...
blitz_idp_mq_latency_bucket{blitz_host="papp01.loc",mq_type="rmq",mq_server="pmq01.
↪loc_5672",broker="blitz.events.direct",op_type="write",le="+Inf",} 3.0
blitz_idp_mq_latency_count{blitz_host="papp01.loc",mq_type="rmq",mq_server="pmq01.
↪loc_5672",broker="blitz.events.direct",op_type="write",} 3.0
blitz_idp_mq_latency_sum{blitz_host="papp01.loc",mq_type="rmq",mq_server="pmq01.
↪loc_5672",broker="blitz.events.direct",op_type="write",} 0.028808135999999998
# HELP blitz_idp_authn_method_app_total Amount of method authentications by app id
# TYPE blitz_idp_authn_method_app_total counter
blitz_idp_authn_method_app_total{blitz_host="papp01.loc",app_id="_blitz_profile",
↪method="sms",status="success",} 2.0
blitz_idp_authn_method_app_total{blitz_host="papp01.loc",app_id="_blitz_profile",
↪method="cls",status="other_error",} 7.0
blitz_idp_authn_method_app_total{blitz_host="papp01.loc",app_id="_blitz_profile",
↪method="password",status="success",} 4.0
blitz_idp_authn_method_app_total{blitz_host="papp01.loc",app_id="_blitz_profile",
↪method="knownDevice",status="other_error",} 3.0
# HELP blitz_idp_authn_method_total Amount of authentications by a method
# TYPE blitz_idp_authn_method_total counter
blitz_idp_authn_method_total{blitz_host="papp01.loc",method="password",status=
↪"success",} 4.0
blitz_idp_authn_method_total{blitz_host="papp01.loc",method="knownDevice",status=
↪"other_error",} 3.0
blitz_idp_authn_method_total{blitz_host="papp01.loc",method="cls",status="other_
↪error",} 7.0
blitz_idp_authn_method_total{blitz_host="papp01.loc",method="sms",status="success",
↪} 2.0
# HELP blitz_idp_authn_method_latency Latency of an authentication method
# TYPE blitz_idp_authn_method_latency histogram
blitz_idp_authn_method_latency_bucket{blitz_host="papp01.loc",method="sms",le="1.0
↪",} 0.0
...
blitz_idp_authn_method_latency_bucket{blitz_host="papp01.loc",method="sms",le="+Inf

```

(continues on next page)

(continued from previous page)

```

↪", } 2.0
blitz_idp_authn_method_latency_count{blitz_host="papp01.loc",method="sms", } 2.0
blitz_idp_authn_method_latency_sum{blitz_host="papp01.loc",method="sms", } 28.
↪6869999999999998
blitz_idp_authn_method_latency_bucket{blitz_host="papp01.loc",method="password",le=
↪"1.0", } 0.0
...
blitz_idp_authn_method_latency_bucket{blitz_host="papp01.loc",method="password",le=
↪"+Inf", } 4.0
blitz_idp_authn_method_latency_count{blitz_host="papp01.loc",method="password", } 4.
↪0
blitz_idp_authn_method_latency_sum{blitz_host="papp01.loc",method="password", }
↪1835.901
# HELP blitz_idp_datasource_connections Amount connections to datasource
# TYPE blitz_idp_datasource_connections gauge
blitz_idp_datasource_connections{blitz_host="papp01.loc",ds_type="ldap",ds_name=
↪"389-ds", } 10.0
# HELP blitz_idp_version Application version
# TYPE blitz_idp_version gauge
blitz_idp_version{blitz_host="papp01.loc",part="major", } 5.0
blitz_idp_version{blitz_host="papp01.loc",part="minor", } 16.0
blitz_idp_version{blitz_host="papp01.loc",part="patch", } 1.0
# HELP blitz_idp_notify_user_total Amount of user notifications by channel
# TYPE blitz_idp_notify_user_total counter
blitz_idp_notify_user_total{blitz_host="papp01.loc",channel="email", } 3.0
blitz_idp_notify_user_total{blitz_host="papp01.loc",channel="sms", } 4.0
blitz_idp_notify_user_total{blitz_host="papp01.loc",channel="push", } 2.0

```

The name of each metric begins with the application name (the hyphen in the name is replaced by an underscore): `blitz_idp_%%,blitz_registration_%%,blitz_recovery_%%,blitz_console_%%`. The list of available metrics is given in the table.

## Blitz Identity Provider performance metrics

Access rights	Type	Description
uptime_seconds	gauge	Time since application start (in seconds)
licence_exp_seconds	gauge	Time until license expires (in seconds)
config_mtime	gauge	Timestamp of configuration file last update
datasource_latency	histogram	Response delays for read and write operations from the account storage ( <code>ldap</code> , <code>jdbc</code> , or <code>couch</code> type)
mq_connections	gauge	Number of connections to MQ ( <code>rmq</code> , <code>kafka</code> )
mq_latency	histogram	Response delays from MQ ( <code>rmq</code> , <code>kafka</code> )
authn_method_app_total	counter	Number of successful and unsuccessful authentications into different applications for each login method
authn_method_total	counter	Total number of successful and unsuccessful authentications for different methods
authn_method_latency	histogram	Authentication duration for different login methods
datasource_connections	gauge	Number of connections to storages
version	gauge	Application version
notify_user_total	counter	Number of notifications sent via different channels
authn_method_app_created	service metrics	These metrics (with the <code>_created</code> suffix) are generated due to Prometheus peculiarities and contain the <code>unix timestamp</code> of the moment the metric was created
authn_method_created		
authn_method_latency_created		
datasource_latency_created		
mq_latency_created		
notify_user_created		

## Using Grafana and Prometheus

For quick setup of monitoring and visualization of Blitz Identity Provider processes, it is convenient to use the Prometheus job assignment and the Grafana dashboard template included in the delivery (`resources.zip`).

**Tip:** The visual representation of data has a wide range of applications. It can be used by managers to analyze workflows, engineers to track situations when the number of authentications exceeds a threshold value (alerts are configured), to monitor the validity of a license, etc. When updating, it is convenient to track the versions of services on a large number of hosts and the time of their launch.

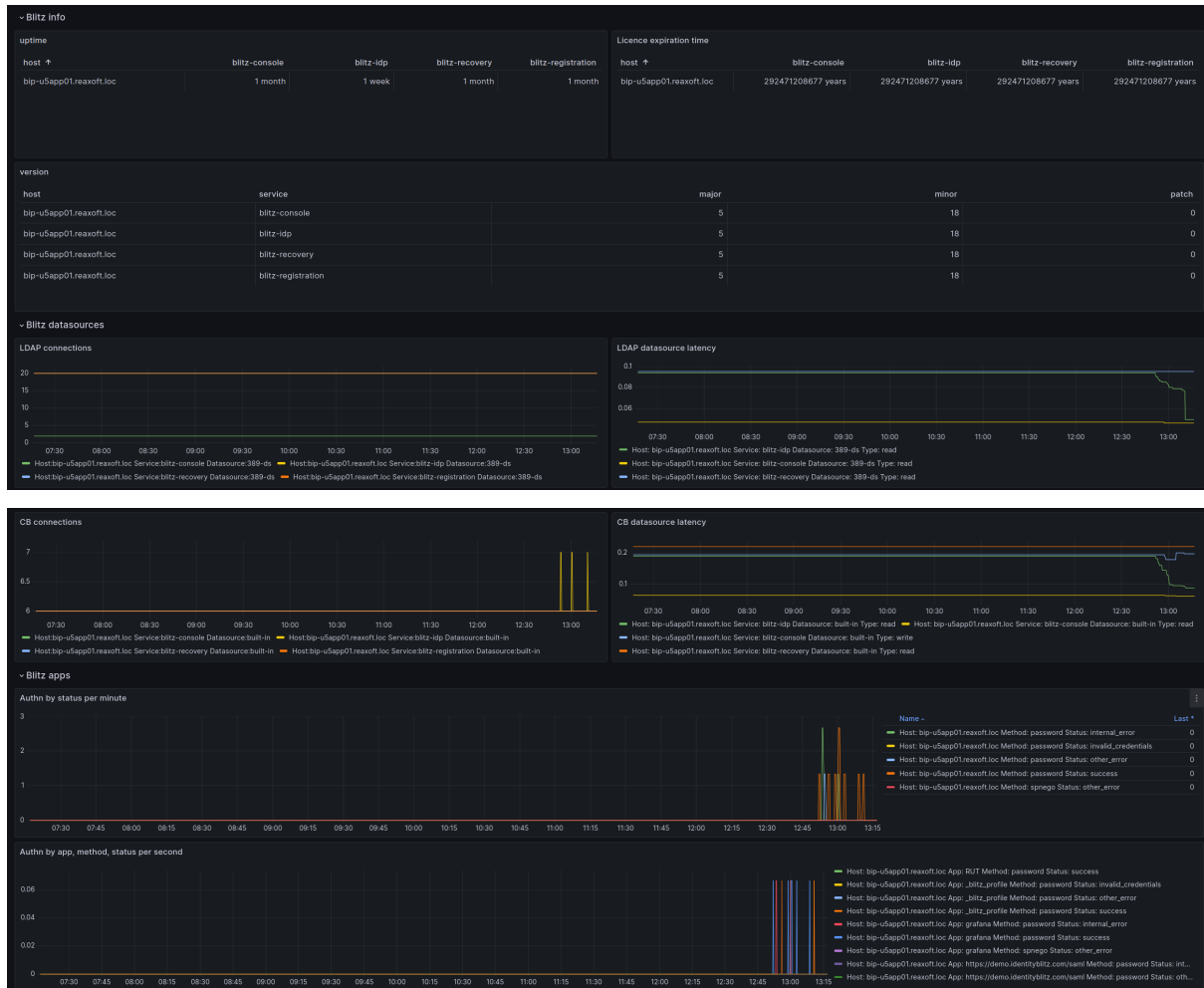
To set up the visualization, follow these steps:

1. Modify the job assignment `prometheus.yaml` according to your system configuration and add it to Prometheus<sup>51</sup>.

<sup>51</sup> <https://prometheus.io>

2. Modify the dashboard template `blitz-dashboard.json`. [Set up Grafana and add a dashboard](#)<sup>52</sup>.

Examples of data visualization in Grafana:



## 2.7.3 Problem solving

Blitz Identity Provider operation logs are written to the `/var/log/identityblitz` directory on each server. The event log of each application is named according to the application:

- `blitz-console.log` - admin console event log;
- `blitz-idp.log` - authentication service event log;
- `blitz-registration.log` - registration service event log;
- `blitz-recovery.log` - access recovery service event log;
- `blitz-keeper.log` - security gateway event log.

When errors related to Blitz Identity Provider operation occur (logged as `[ERROR]`), it is recommended to contact Blitz Identity Provider technical support at [support@idblitz.com](mailto:support@idblitz.com). When contacting Blitz Identity Provider, please specify the version of Blitz Identity Provider you are using.

If you need to change the logging level, you need to change the logging levels in the `blitz.conf` configuration file in the `logger` block.

The following logging levels are set by default:

<sup>52</sup> <https://prometheus.io/docs/visualization/grafana/>



```

"levels" : {
  "ROOT" : "TRACE",
  "application" : "TRACE",
  "com.couchbase.client" : "INFO",
  "com.couchbase.service" : "INFO",
  "com.couchbase.endpoint" : "INFO",
  "com.couchbase.node" : "INFO",
  "com.couchbase.tracing" : "INFO",
  "com.identityblitz" : "TRACE",
  "com.identityblitz.idp" : "TRACE",
  "com.identityblitz.idp.events" : "TRACE",
  "com.identityblitz.idp.flow.dynamic" : "TRACE",
  "com.identityblitz.idp.flow.dynamic.extend" : "TRACE",
  "com.identityblitz.idp.task.processing" : "DEBUG",
  "com.identityblitz.login-framework" : "TRACE",
  "com.identityblitz.login-framework.ldap-timings" : "INFO",
  "com.identityblitz.login.store" : "TRACE",
  "com.identityblitz.idp.rabbitmq" : "INFO",
  "com.identityblitz.play.memcached" : "INFO",
  "com.identityblitz.play.memcached.RefreshableMemcachedConnection" : "INFO",
  "com.unboundid.ldap.sdk" : "TRACE",
  "org.asynchttpclient.netty" : "TRACE",
  "org.opensaml" : "INFO",
  "org.opensaml.util.resource" : "INFO",
  "play" : "TRACE",
  "plugin.memcached" : "INFO"
}

```

To change the logging level, the `ROOT` and all `com.identityblitz.*` parameters should be assigned the value `TRACE`.

If the Blitz Identity Provider configuration change was accidentally made in the admin console, the previous versions of the `blitz.conf` and `console.conf` configuration files are saved in the hidden `/usr/share/identityblitz/blitz-config/.snapshot` directory. You can use these files to roll back to a previous configuration or to determine differences with the current configuration files.

To find out at what time and by whom a configuration file was changed, comments are placed at the beginning of the `blitz.conf` and `console.conf` configuration files indicating the time of editing and the author of the changes. An example of an audit record of a configuration file change is given below:

```

#####
↪#####
# modified: 2021-05-09 20:55:55 MSK
# author: admin
# ip: 0:0:0:0:0:0:1
# user agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
#####
↪#####

```

### 2.7.4 Security gateway

*Blitz Keeper* (page 451) is a separately installable module that is used as the Blitz Identity Provider security gateway.

# Chapter 3

## Integration

### 3.1 Preparing for integration

#### 3.1.1 Selecting an interaction protocol

When integrating the application with Blitz Identity Provider, one of the interaction protocols should be selected to identify and authenticate the user:

- [OpenID Connect 1.0 \(OIDC\)<sup>53</sup> / OAuth 2.0<sup>54</sup>](#) is a modern SSO protocol, initially focused on working with web and mobile applications on the Internet.

---

**Tip:** If a new application is being created, it is recommended to connect it to Blitz Identity Provider using [OIDC/OAuth 2.0](#).

---

- [SAML 1.0/1.1/2.0<sup>55</sup>](#) is an SSO protocol that allows you to connect various enterprise software or cloud applications to the login service.

**Attention:** The connected application must have built-in SAML support, or such support can be added as an additional option or through the installation of an integration connector/plugin.

The choice of protocol largely depends on which application you want to connect:

- if the application supports one of the SSO protocols, then it is worth connecting it using this protocol;
- if the proposal does not support protocols, then it should be finalized – in this case, it is recommended to support OIDC interaction;
- if the application is just being created, then at this stage it is advisable to support one of the SSO protocols - it is easier to implement OIDC support, however, when using the available SAML libraries, this protocol can also be used.

The table below shows some of the features of the OIDC and SAML protocols.

---

<sup>53</sup> [https://openid.net/specs/openid-connect-core-1\\_0.html](https://openid.net/specs/openid-connect-core-1_0.html)

<sup>54</sup> <https://tools.ietf.org/html/rfc6749>

<sup>55</sup> <https://www.oasis-open.org/standards#samlv2.0>

## Features of connection protocols

	OIDC/OAuth 2.0	SAML 1.0/1.1/2.0
A way to ensure trust between the application and Blitz Identity Provider	The secret of the application (usually in the form of a string), known as Blitz Identity Provider	Electronic signature. Both authentication requests and responses are signed XML documents
Interaction method	Authentication takes place through the user's web browser. To complete authentication, the backend of the application must generate an HTTP request to Blitz Identity Provider	Usually, the authentication request and response go through the user's web browser. The application and Blitz Identity Provider may not have network connectivity
Getting user information	Two ways to get user data: <ul style="list-style-type: none"> <li>• The application accesses the Blitz Identity Provider REST service and receives user data in JSON format. The application can continue to receive user data even when the user ends their online session</li> <li>• The application receives user data from the identification token (id_token in the JWT form) received from Blitz Identity Provider based on the login results</li> </ul>	The user data is contained in the response to the authentication request in XML format. The application can receive data from Blitz Identity Provider only at the time of user login
Supported applications	Web and mobile applications	Web applications

**Note:** OIDC allows you to implement all the basic SAML scenarios, but it uses a simpler JSON/REST protocol. A significant advantage of OIDC is the support of mobile applications.

**Important:** If the application connected to Blitz Identity Provider cannot be finalized, but the application is a web application deployed in its own infrastructure (on-premise), then you can connect the application to Blitz Identity Provider using a web proxy and the *Simple protocol* (page 181) specially implemented in Blitz Identity Provider.

## 3.2 OIDC application integration

### 3.2.1 How to register the application correctly

Authentication in OIDC/OAuth terminology 2.0 is the result of the interaction of three parties:

- the authorization service (Authorization Server) or the resource provider (Resource Server), which is Blitz Identity Provider;
- the client system (Client), which is an application that requests access to a resource (user information and data);
- the resource owner (Resource Owner), which is the user, since during authentication he allows access to data about himself.

The first step when connecting an application is to [register it](#) (page 171) as a client system in Blitz Identity Provider. Authentication requests will use and take into account the data specified during application registration:

### Web application

- application ID (`client_id`);
- application secret (`client_secret`).
- permitted return addresses (lists `redirect_uri` and `post_logout_redirect_uri`);
- list of requested permissions (`scope` list);
- information about non-standard modes required by the application:
  - the application needs to receive a `refresh_token` – by default, the `refresh_token` application will not be returned; when selecting this mode, you must additionally specify the required validity period of `refresh_token` (by default, the validity period of the token will be 1 day, the maximum possible - 365 days);
  - the application needs to use a non-standard interaction scenario (for example, Implicit Flow, Hybrid Flow) - by default, the application is allowed to use only Authorization Code Flow;
  - the application needs to receive an access token in the JWT format – by default, the access token is provided in the `opaque` format;
  - the application needs to receive an access token (`access_token`) with a non-standard expiration date - the access token is valid for 1 hour as standard;
- a list of additional attributes that Blitz Identity Provider should add to the identification token (additional attributes to be passed as part of `id_token`);
- login mode (login as an individual or as a representative of an organization).

### Mobile application

- mobile application ID (`software_id`);
- initial access token (Initial Access Token);
- application metadata in the form of a JWS token (`software_statement`).
- permitted return addresses (lists `redirect_uri` and `post_logout_redirect_uri`);
- list of requested permissions (`scope` list);
- non-standard modes required by the application:
  - the application needs to receive a `refresh_token` – by default, the `refresh_token` application will not be returned; when selecting this mode, you must additionally specify the required validity period of `refresh_token` (by default, the validity period of the token will be 1 day, the maximum possible - 365 days);
  - the application needs to use a non-standard interaction scenario (for example, Implicit Flow, Hybrid Flow) - by default, the application is allowed to use only Authorization Code Flow;
  - the application needs to receive an access token in the JWT format – by default, the access token is provided in the `opaque` format;
  - the application needs to receive an access token (`access_token`) with a non-standard expiration date - the access token is valid for 1 hour as standard;
- a list of additional attributes that Blitz Identity Provider should add to the identification token (additional attributes to be passed as part of `id_token`);

- login mode (login as an individual or as a representative of an organization).

**Note:** When developing a mobile application, you can use both common `Initial Access Token` and `software_statement` for your iOS/Android implementations, and request different sets of `Initial Access Token` and `software_statement` for each OS and possibly each edition (phone/tablet) and even the version of the application. For simplicity of further presentation, the text of the document will imply that the mobile application uses one common `Initial Access Token` and one common `software_statement`.

When creating a login function in mobile applications using Blitz Identity Provider, it is recommended to take into account the following features:

- it is inconvenient for mobile app users to enter their username and password on the authentication Blitz Identity Provider web page every time they log in. Instead, they are more accustomed to using the PIN code of the application or Touch ID/Face ID when re-logging in;
- a user can use his/her Blitz Identity Provider account to log in to multiple installations of the same mobile application (for example, log in to an application installed on an iPhone and log in to the same application installed on an iPad). The user should be able to revoke the access rights granted to these application installations to their information in Blitz Identity Provider;
- for security reasons, it is undesirable to store the application password (`client_secret`) on the user's device (inside the mobile application assembly), which is used to interact the application with Blitz Identity Provider.

To take into account the above features, Blitz Identity Provider provides a number of special mechanisms designed for use by mobile applications.

The recommended scenario for the interaction of a mobile application with Blitz Identity Provider is described in [Connecting a mobile app](#) (page 317).

Below you will find information on how to determine which allowed return addresses, `scope` permissions, additional attributes in `id_token` you can set when registering the application in Blitz Identity Provider.

### How to determine the return addresses

The request for user identification/authentication contains a return link during authorization (`redirect_uri`), where the user should be returned after passing identification/authentication. Valid return links must match the allowed prefixes registered in Blitz Identity Provider.

If the return link is specified in the identification/authentication request and it does not match any of the specified prefixes, then identification/authentication will be refused.

Depending on the type of connected application, it is recommended to use the following return link prefixes:

- When connecting web applications, the application domain names should be used as the prefixes of the return links. For example, if after authentication it is required to return the user to `https://domain.com/callback`, then the prefix of the return link should be `https://domain.com/`.

**Warning:** When connecting to the Blitz Identity Provider production environment, the web application should use only HTTPS handlers as `redirect_uri` and `post_logout_redirect_uri`. Using HTTP to interact with the Blitz Identity Provider production environment is prohibited.

- When connecting mobile applications, it is recommended to specify the return links themselves as prefixes of one of the types: links of the `private-use URI scheme` type (for example, `com.example.app:/oauth2redirect/example-provider` type) or links of the `Universal links` type (for example, `https://app.example.com/oauth2redirect/example-provider`).

**Note:** Links like `Universal links` are available starting from iOS 9 and Android 6.0 and are preferred for use. It is recommended to use the `private-use` URI scheme links only if the application should run on earlier versions of iOS/Android.

The logout request contains a return link during the logout (`post_logout_redirect_uri`). This link indicates where the user should be returned after a successful logout. Valid return links must match the allowed prefixes registered in Blitz Identity Provider (the prefix must contain the domain name of the application and part of the path, at least, `https://domain.com/`). If a return link is specified in the logout request and it does not match any of the specified prefixes, an error will be displayed.

### What permissions can be requested

Permissions (`scope` in OIDC/OAuth 2.0 terminology) determine which data and which rights to perform which operations the application will receive based on authentication results.

The list of permissions provided in Blitz Identity Provider is shown in the table.

### Available permissions (scope)

Permission	Description	Composition of the received attributes
<code>openid</code>	A technical authorization indicating that authentication is performed according to the OIDC specification	When requesting this <code>scope</code> , Blitz Identity Provider provides the application with an <code>id_token</code> . From the <code>id_token</code> the application can get the necessary <i>user attributes</i> (page 309).
<code>profile</code>	Basic user profile data	List of data: <ul style="list-style-type: none"> <li>• <code>sub</code> is a unique identifier</li> <li>• <code>family_name</code> is a surname</li> <li>• <code>given_name</code> is a given name</li> <li>• <code>middle_name</code> is a middle name</li> <li>• <code>email</code> is a business email address</li> <li>• <code>phone_number</code> is a mobile phone number</li> </ul>
<code>usr_grps</code>	Getting a list of user groups	<code>groups</code> is a list of groups that the user is included in. Each entry in the list includes the following attributes of the organization: <ul style="list-style-type: none"> <li>• <code>id</code> is the ID of the group</li> <li>• <code>name</code> is the name of the group</li> </ul>
<code>native</code>	Permission to perform end-to-end login to the web application from the mobile application	Relevant only for <i>mobile applications</i> (page 324).

### What additional attributes can be included in the `id_token`

There is usually no need to get user attributes directly from the identification token (`id_token`) – a simpler and recommended way is to *get user data* (page 333) through a REST service call.

If you still need to get information about the user from `id_token` (page 309), then the available attributes are selected from the following list.

### Possible additional user attributes in id\_token

Attribute	Description
family_name	Last name
given_name	Name
middle_name	Patronymic
email	E-mail address
phone_number	Mobile phone

**Tip:** Blitz Identity Provider also allows you to place application design elements on the Blitz Identity Provider login page. If you want to create a personalized login page for the connected system, you need to adapt the template for the design of the login page to the design of the connected system. The template for the design of the login page is a zip archive, inside which the HTML framework of the login page and the stylesheet, images, and JavaScript handlers used on the page are recorded.

The prepared archive of the login page theme should be *uploaded* (page 221) to Blitz Identity Provider.

### 3.2.2 Connecting a web application

**Tip:** See the *description* (page 162) of the interaction between a web application and Blitz Identity Provider via OIDC.

#### Connection settings

To connect a mobile application to Blitz Identity Provider, you will need the data obtained when *registering it in product* (page 295):

- the identifier assigned to the application in Blitz Identity Provider (`client_id`);
- the secret of the application (`client_secret`);
- return URLs registered for the application during authorization;
- logout return URLs registered for the application;
- the permissions registered for the application (`scope`).

In order to interact with Blitz Identity Provider, the web application must use the following addresses:

- URL for authorization and authentication:
  - `https://login-test.company.com/blitz/oauth/ae` (test environment)
  - `https://login.company.com/blitz/oauth/ae` (production environment)
- URL for getting and updating the access token:
  - `https://login-test.company.com/blitz/oauth/te` (test environment)
  - `https://login.company.com/blitz/oauth/te` (production environment)
- URL for getting user data:
  - `https://login-test.company.com/blitz/oauth/me` (test environment)
  - `https://login.company.com/blitz/oauth/me` (production environment)
- URL for getting access token data:
  - `https://login-test.company.com/blitz/oauth/introspect` (test environment)



- `https://login.company.com/blitz/oauth/introspect` (production environment)
- URL for performing the logout:
  - `https://login-test.company.com/blitz/oauth/logout` (test environment)
  - `https://login.company.com/blitz/oauth/logout` (production environment)

All these URLs, as well as additional information, are located at the address of dynamically updated settings (metadata) of each Blitz Identity Provider environment:

---

**Tip:** See [RFC 8414 OAuth 2.0 Authorization Server Metadata](#)<sup>56</sup>.

---

- `https://login-test.company.com/blitz/.well-known/openid-configuration` (test environment)
- `https://login.company.com/blitz/.well-known/openid-configuration` (productive environment)

Application developers can use a single link to Blitz Identity Provider metadata instead of listing all of the URLs in their application's configuration.

### Ready-made libraries

To integrate an application with Blitz Identity Provider, you can use one of the many [ready-made OAuth 2.0 libraries](#)<sup>57</sup> or implement the interaction yourself.

### Getting the authorization code

To identify and authenticate the user, the application must direct the user to the URL to receive the authorization code in Blitz Identity Provider, passing as parameters:

- `client_id` is the client's ID;
- `response_type` – response type (takes the value `code`, `token`, `code token`, `code id_token`, `code id_token token`, `id_token token`, `id_token`);

---

**Important:** The value of the `response_type` parameter indicates the way the application has chosen to interact with Blitz Identity Provider:

- `code` – Authorization Code Flow;
  - `code token`, `code id_token token`, `code id_token token` – Hybrid Flow;
  - `id_token token`, `id_token` – OIDC Implicit Flow;
  - `token` – OAuth 2.0 Implicit Flow.
- 

- `response_mode` (optional parameter) – allows you to explicitly specify the required method of transmitting the authorization code. When the application is normally connected to Blitz Identity Provider, this parameter should not be transmitted, since it is recommended to use standard methods of transmitting the authorization code (`query` – for Authorization Code Flow and `fragment` – for *Implicit/Hybrid Flow*).

Possible values of the `response_mode` parameter:

- `query` – the value of the authorization code (`code`) is returned to the `redirect_uri` of the application in the form of a query parameter. The standard mode for Authorization Code Flow.

---

<sup>56</sup> <https://tools.ietf.org/html/rfc8414>

<sup>57</sup> <https://oauth.net/code/#client-libraries>

- `fragment` – the value of the authorization code (code) is returned to the `redirect_uri` of the application in the form of a fragment parameter (#). The standard mode for Implicit Flow.
- `form_post` – in this mode, the authorization response parameters are encoded as HTML form values, which are automatically sent to the User Agent and transmitted to the client via the HTTP POST method, while the resulting parameters are encoded in the body using the `application/x-www-form-urlencoded` format.
- `scope` – the requested permissions, for authentication, the `openid` permission and the necessary additional scope must be passed to receive user data, for example, `profile` (when multiple scopes are requested, they are transmitted in one line and separated from each other by a space);
- `redirect_uri` is a link to return the user to the application, the link must match one of the registered values;
- `state` is a set of random characters in the form of a 128-bit request identifier (used to protect against interception), the same value will be returned in the response – an optional parameter;
- `access_type` (optional parameter) – whether the application needs to receive `refresh_token`, which is necessary to obtain information about the user in the future when the user is offline. Takes the value `online` or `offline`, `refresh_token` is provided when `access_type=offline`. If the value is not set, then the behavior is determined by the setting set for the specified application in Blitz Identity Provider;
- `prompt` (optional parameter) – specifies Blitz Identity Provider the required login mode. Possible values of the `prompt` parameter:
  - `none` is a ban on authentication.

If, when executing a request, Blitz Identity Provider needs to display the identification/authentication request screen to the user, Blitz Identity Provider will not do this, but will return the `login_required` error to the system on its `redirect_uri`. A call with the `prompt=none` parameter should be made if the application wants to check if the user has a Blitz Identity Provider session, but does not want the user to see the Blitz Identity Provider login screen when performing such a check.

- `select_account` – request to change the current user.

Blitz Identity Provider will display an account selection screen to the user so that the user can log in with a different account.

- `login` is a ban on SSO.

If, when executing the request, Blitz Identity Provider finds out that the user has already been identified/authenticated before, then Blitz Identity Provider will explicitly require the user to re-identify/authenticate. At the same time, Blitz Identity Provider additionally checks that the login will be performed by the same user whose user session is open.

If, during re-identification/authentication, the user logs in with a different account, Blitz Identity Provider returns the `login_required` error to the system on its `redirect_uri`. A call with the `prompt=login` parameter should be made if the application wants to explicitly request identification/authentication from the user, for example, when accessing an application function that requires increased protection.

**Note:** For `prompt=login` for the application, if necessary, you can enable a different scenario for processing the situation that the user logged in with a different account than he was previously logged in to the session. Namely, you can enable that when `prompt=login` is called, the current session is forcibly logged out and the session is created under a new account. This behavior is not recommended, but can be enabled for the application on a separate request.

- `nonce` (optional parameter) is a string used to bind an application session to an identification token. When the application is connected to Blitz Identity Provider using Authorization Code Flow as standard, there is no need to use the `nonce` parameter.

When connecting via Implicit Flow or Hybrid Flow, this parameter must be passed. The `nonce` value must be a random text string.

- `display` (optional parameter) – the parameter in the `script` value is passed only if the login process is started via [HTTP API](#) (page 432).
- `bip_action_hint` (optional parameter) – specifies Blitz Identity Provider that the login page should open in one of the special modes:
  - `open_reg` – open in user registration mode; when using this mode, you can additionally specify the `login_hint` parameter with the user’s email value, and then the “Email address” field will be filled with the specified email value;
  - `open_recovery` – open in password recovery mode; when using this mode, you can additionally specify the `login_hint` parameter with the value of the user’s email, and then the “Login” field will be filled with the specified email value;
  - `used_externalIdps:apple:apple_1` – open in Apple ID login mode;
  - `used_externalIdps:facebook:facebook_1` – open in Facebook login mode<sup>1</sup>;
  - `used_externalIdps:google:google_1` – open in Google login mode;
  - `used_password` – open in password login mode (default behavior);
  - `used_webAuthn` – open in login mode using the FIDO2 key (Passkey);
  - `used_x509` – open in the login mode by electronic signature;
  - `used_qrCode` – open in QR code login mode;
  - `used_spnego` – open in login mode by operating system session;
  - `used_sms` – open in the login mode by SMS code;
  - `used_outside_methodname` – open in login mode via an external authentication method named `methodname`.
- `bip_user_hint` (optional parameter) – the identifier (sub) of the user account is passed, which should be selected automatically when the login screen is opened.

The ID must match one of the accounts stored on the device, or the login page will be opened in the new user login mode;

- `login_hint` (optional parameter) – a value is passed that must be filled in the login field if the login page is open in the new user login mode.

If you need to fill in the login in the case when there is already a remembered user, then you need to use the `login_hint` parameter in combination with the `bip_user_hint` parameter;

- `bip_extIdps_user_choose_hint` (optional parameter) – the identifier (sub) of the user account is passed, which should be selected automatically if the user logs in through an external identification provider to which several Blitz Identity Provider accounts are linked;
- `code_challenge_method` (optional parameter) – the value “S256” is passed if the connected application supports the PKCE specification for additional protection of interaction with Blitz Identity Provider.

---

**Tip:** See [RFC 7636 Proof Key for Code Exchange by OAuth Public Clients](#)<sup>58</sup>.

---

PKCE is not required to connect web applications.

PKCE must be used to connect mobile applications to Blitz Identity Provider.

---

<sup>1</sup> Meta is recognized as an extremist organization and is banned in Russia, while the activities of its social networks Facebook and Instagram are also banned in Russia.

<sup>58</sup> <https://tools.ietf.org/html/rfc7636>

- `code_challenge` (optional parameter) – when using PKCE, the value calculated from `code_verifier` is passed to this parameter using the following formula:

**Tip:** When debugging, it is convenient to use the [online calculator](#)<sup>59</sup>.

```
code_challenge=BASE64URL-ENCODE(SHA256(ASCII(code_verifier)))
```

**Note:** It is forbidden to open the login page in the frame. The user should see the URL of the login page, and also be able to verify that the HTTPS connection is available by the web portal “login.company.com”.

Example of a request to receive an authorization code (identification/authentication and access token with `openid` and `profile` permissions were requested):

```
https://login.company.com/blitz/oauth/ae?client_id=ais&response_type=code&
↳scope=openid+profile&access_type=offline&state=342a2c0c-d9ef-4cd6-b328-
↳b67d9baf6a7f&redirect_uri=https%3A%2F%2Fapp.company.com%2Fre
```

An example of a response with the value of the authorization code (`code`) and the `state` parameter:

```
https://app.company.com/re?code=f954...nS0&state=342a2c0c-d9ef-4cd6-b328-b67d9baf6a7f
```

Possible errors when calling `/oauth/ae` comply with RFC 6749 and are described [here](#)<sup>60</sup>.

An example of a request for an authorization code in which Blitz Identity Provider should not open the login page if the user has not yet been identified/authenticated in the current web browser:

```
https://login.company.com/blitz/oauth/ae?client_id=ais&response_type=code&
↳scope=openid+profile&access_type=offline&state=342a2c0c-d9ef-4cd6-b328-
↳b67d9baf6a7f&prompt=none&redirect_uri=https%3A%2F%2Fapp.company.com%2Fre
```

An example of an error response if, in order to receive the authorization code, the user must explicitly pass identification/authentication on the Blitz Identity Provider login page, and the request was executed with the `prompt=none` parameter:

```
https://app.company.com/re?error=login_required&error_
↳description=The+Authorization+Server+requires+End-User+authentication...&
↳state=342a2c0c-d9ef-4cd6-b328-b67d9baf6a7f
```

Example of a request for an access token and an identification token using OIDC Implicit Flow:

```
https://login.company.com/blitz/oauth/ae?client_id=ais&response_type=id_token
↳%20token&scope=openid+profile&state=342a2c0c-d9ef-4cd6-b328-b67d9baf6a7f&nonce=n-
↳0S6_WzA2Mj&redirect_uri=https%3A%2F%2Fapp.company.com%2Fre
```

An example of a response from Blitz Identity Provider with access and identification tokens obtained using OIDC Implicit Flow:

```
https://app.company.com/re#access_token=SlAV32hkKG&token_type=Bearer&id_
↳token=eyJ0...NiJ9.eyJ1c...I6IjIifX0.DeWt4Qu...ZXso&expires_in=3600&state=342a2c0c-d9ef-
↳4cd6-b328-b67d9baf6a7f
```

Example of a request for an authorization code and an identification token using OIDC Hybrid Flow:

<sup>59</sup> <https://example-app.com/pkce>

<sup>60</sup> <https://tools.ietf.org/html/rfc6749#section-4.1.2.1>

```
https://login.company.com/blitz/oauth/ae?client_id=ais&response_type=code%20id_
↪token&scope=openid+profile&state=342a2c0c-d9ef-4cd6-b328-b67d9baf6a7f&nonce=n-
↪0S6_WzA2Mj&redirect_uri=https%3A%2F%2Fapp.company.com%2Fre
```

An example of a response from Blitz Identity Provider with access and identification tokens obtained using OIDC Hybrid Flow:

```
https://app.company.com/re#code=f954...FxS0&id_token=eyJ0...NiJ9.eyJ1c...I6IjIifX0.
↪DeWt4Qu...ZXso&state=342a2c0c-d9ef-4cd6-b328-b67d9baf6a7f
```

## Getting tokens

In order to carry out the result of identification/authentication of the user and obtain his data, Blitz Identity Provider issues various tokens to the application.

### Tokens used in Blitz Identity Provider

Name	Designation	Purpose and validity period
Access token	access_token	Getting access to a protected resource, for example, user data. The token is valid for 3600 seconds.
Refresh token	refresh_token	Updating the access token. The refresh_token token is provided only if the application specified the need to receive a refresh_token during registration, or if the access_type=offline parameter was specified in the request for an authorization code. The token is valid until the moment of use, but no longer than 365 days.
ID token	id_token	Obtaining identification information, for example, a user ID. The token is valid for 3 hours.

### Exchange of the authorization code for tokens

After receiving the authorization code, the application must exchange it for tokens.

**Attention:** The token collection service must be called from the servers of the application connected to Blitz Identity Provider. Calling the service from the program code executed on the side of the web browser (for example, from the JavaScript code of a web page) is **PROHIBITED**. The received access token (access\_token) must be processed by the backend of the application and must not be transmitted through the user's browser.

Method POST `https://login.company.com/blitz/oauth/te`

Headers Authorization with the value Basic {secret}, where secret is client\_id:client\_secret (for example, app:topsecret) in Base64 format.

Request body

- code – the value of the authorization code that was previously received;
- grant\_type – takes the value authorization\_code, if the authorization code is exchanged for an access token;

- `redirect_uri` – the link to which the user should be directed after giving permission for access (the same value that was specified in the request for an authorization code);
- `code_verifier` (only if PKCE is used) is the value of the verification code used in calculating the `code_challenge` when receiving the authorization code.

#### Returns

- If successful, an access token, an update token, and an identification token.

---

**Tip:** Using the received access token, the application can [request](#) (page 333) up-to-date user data from Blitz Identity Provider.

---

- If the authorization code has already been used, the `redirect_uri` did not match the one previously used in the call to `/oauth/ae`, or the code expired, or the `code_verifier` passed does not match `code_challenge`, an error will be returned as a response. Possible errors when calling `/oauth/te` comply with RFC 6749 and are described [here](#)<sup>61</sup>.

### Examples

#### Request

```
POST /blitz/oauth/te HTTP/1.1
Authorization: Basic cG9ydGFsLmlhc2l1LmxvY2FsOnBvcnRhbcC5pYXNpdS5sb2NhbA==
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&code=FLZHS...GU&redirect_uri=https%3A%2F%2Fapp.company.
↪com%2Fre
```

#### Response

```
{
  "id_token": "eyJhbGciOiJSUzI1NiJ9.eyJub30=.Ckt...sQ",
  "access_token": "dO-xym...BE",
  "expires_in": 3600,
  "refresh_token": "11EWX...Iw",
  "token_type": "Bearer"
}
```

#### Error

```
{
  "error": "invalid_grant",
  "error_description": "The provided authorization grant ... is invalid, expired, ↪
↪revoked..."
}
```

<sup>61</sup> <https://tools.ietf.org/html/rfc6749#section-5.2>

## Updating the access token

Method POST `https://login.company.com/blitz/oauth/te`

Headers Authorization with the value `Basic {secret}`, where `secret` is `client_id:client_secret` (for example, `app:topsecret`) in Base64 format.

Request body

- `refresh_token` is refresh token;
- `grant_type` – takes the value “refresh\_token” if the refresh token is exchanged for an access token.

Listing 1: Request example

```
POST /blitz/oauth/te HTTP/1.1
Authorization: Basic cG9ydGFsLmlhc2l1LmxvY2FsOnBvcnRhbcC5pYXNpdS5sb2NhbA==
Content-Type: application/x-www-form-urlencoded

grant_type=refresh_token&refresh_token=jj2DA...bQ
```

## Exchanging an access token

An application can exchange an `access_token` with one set of permissions (`scopes`) and claims (`claims`) for an `access_token` with another set of permissions and claims using the [OAuth 2.0 Token Exchange](#)<sup>62</sup>. This can be useful before transferring the `access_token` from the application that received it to another application, so that the application receives a reduced set of permissions and user information.

**Attention:** To use the access token exchange, the application must be granted special permission to use the OAuth 2.0 Token Exchange (`urn:ietf:params:oauth:grant-type:token-exchange` is allowed). The settings for the access token exchange rules must also be set.

Method POST `https://login.company.com/blitz/oauth/te`

Headers Authorization with the value `Basic {secret}`, where `secret` is `client_id:client_secret` (for example, `app:topsecret`) in Base64 format.

Request body

**Attention:** One of the `resource` or `audience` parameters must be specified.

- `grant_type` – takes the value `urn:ietf:params:oauth:grant-type:token-exchange`.
- `resource` – takes the name of the resource for which the exchange of the access token is requested.
- `audience` – takes the names of applications for which an access token is requested.
- `subject_token_type` – the required type of the received token is passed. In the current version of Blitz Identity Provider, only the `urn:ietf:params:oauth:token-type:access_token` type is supported.
- `subject_token` – the value of the replaced access token (`access_token`) is passed.
- Optional parameter `scope` – specifies the list of requested `scope` in the new token. If this parameter is not specified, then all the `scope` allowed by the exchange rule will be included in the new token.
- Optional parameter `token_format` – specifies the required format for the issued access token. Possible values: `jwt` or `opaque`. If this parameter is omitted, the new access token will be issued in the same format as the access token passed to `subject_token`.

<sup>62</sup> <https://www.rfc-editor.org/rfc/rfc8693.txt>

## Examples

### Request

Listing 2: Standard request

```
POST /blitz/oauth/te HTTP/1.1
Authorization: Basic cG9...A==
Content-Type: application/x-www-form-urlencoded

grant_type=urn:ietf:params:oauth:grant-type:token-exchange&resource=...&subject_
↔token_type=urn:ietf:params:oauth:token-type:access_token&subject_token=eyJ...vA
```

Listing 3: Request with transmission of audience, token\_format and scope

```
POST /blitz/oauth/te HTTP/1.1
Authorization: Basic cG9...A==
Content-Type: application/x-www-form-urlencoded

grant_type=urn:ietf:params:oauth:grant-type:token-exchange&token_format=opawue&
↔audience=system1 system2&scope=openid profile&subject_token_
↔type=urn:ietf:params:oauth:token-type:access_token&subject_token=uuy...OE
```

### Response

```
{
  "access_token": "eyJr...-g",
  "expires_in": 3600,
  "scope": "openid new_scope",
  "token_type": "Bearer",
  "issued_token_type": "urn:ietf:params:oauth:token-type:access_token"
}
```

### Error

Listing 4: No rules were found to allow the requested access token exchange

```
{
  "error": "invalid_target",
  "error_description": "Access denied for resource or audience"
}
```



Listing 5: The access token has expired

```
{
  "error": "bad_access_token",
  "error_description": "Access token 'CmJ...Dk' not found"
}
```

### Using OAuth 2.0 Resource Owner Password Credentials

If the application has been granted special permission to use OAuth 2.0 Resource Owner Password Credentials (ROPC) (*grant\_type = password* is allowed), then the application can request an access token as follows.

Method `POST https://login.company.com/blitz/oauth/te`

Headers `Authorization` with the value `Basic {secret}`, where `secret` is `client_id:client_secret` (for example, `app:topsecret`) in Base64 format.

Request body

- `grant_type` – takes the value `password`;
- `username` – contains the username of the user;
- `password` – contains the user's password;
- `scope` – contains a list of requested permissions.

Returns

- If successful, an access token.
- In case of failure, an error. Possible values for `error_description` in case of an account problem:
  - `Invalid user credentials` – invalid username or password;
  - `User_locked` – account locked;
  - `User locked by inactivity` – the account is blocked due to prolonged inactivity;
  - `Password method locked` – a ban on using password authentication is enabled for the account;
  - `Password method not configured` – the password authentication method is not configured;
  - `Password expired` – the password has expired;
  - `Need password change` – a mandatory password change is required when logging in.

### Request

```
POST /blitz/oauth/te HTTP/1.1
Authorization: Basic cG9...A==
Content-Type: application/x-www-form-urlencoded

grant_type=password&username=testuser&password=testpwd1&scope=profile
```

## Response

```
{
  "access_token": "dO-xym...BE",
  "expires_in": 3600,
  "scope": "profile",
  "token_type": "Bearer"
}
```

## Error

```
{
  "error": "invalid_grant",
  "error_description": "Invalid user credentials"
}
```

## ID token

To obtain identification and authentication data, the application can independently analyze the content of the ID token (`id_token`).

**Tip:** Instead of analyzing the `id_token`, it is recommended to use a request to [updating user data](#) (page 333) by access token.

**Token structure** The ID token consists of three parts:

- the header (`header`), which contains general information about the type of token, including the cryptographic operations used during its formation;
- a set of statements (`payload / claim set`) with meaningful information about the token;
- a signature (`signature`) that certifies that the token was issued by Blitz Identity Provider and was not changed during transmission.

The parts of the token are separated by a dot, it looks like:

```
HEADER.PAYLOAD.SIGNATURE
```

The token is passed as a string in the `Base64url` format.

**Token header**

- `alg` – description of the encryption algorithm (parameter `alg`); currently Blitz Identity Provider supports the electronic signature algorithm `RSA-SHA-256` recommended by the specification (corresponds to the value `RS256`);
- `kid` – ID of the key used to sign the token.

**Set of statements** Attributes:

- `exp` is the time of expiration, indicated in seconds from January 1, 1970. 00:00:00 GMT;
- `iat` is the time of issue, indicated in seconds from January 1, 1970. 00:00:00 GMT;
- `sub` is the identifier of the subject, the value of the user ID is specified as the value;
- `ua_id` is the user's device identifier;
- `aud` is the recipient of the token, the `client_id` of the application that sent the authentication request is indicated;

- `iss` – the organization that issued the token, specify the issuer URL, `https://login.company.com/blitz` by default;
- `nonce` is a security string, the value `nonce` is specified, which was passed by the application to Blitz Identity Provider in the original request to `/oauth/ae`. It is used only with Implicit or Hybrid Flow. When an application receives a token using Implicit or Hybrid Flow, the application must match `nonce` from the identification token with `nonce` from its request;
- `at_hash` is half of the hash of the access token, transmitted only when using Implicit or Hybrid Flow. It represents the Base64 encoded left half of the value of the SHA-256 function from `access_token`. An application that receives an access token using Implicit or Hybrid Flow must extract the value `at_hash` from the identification token and compare it with the access token.
- `c_hash` is half of the hash of the authorization code, transmitted only if Hybrid Flow is used. It is the Base64 encoded left half (128 bits) of the SHA-256 function value from the authorization code (`code`); An application that receives an authorization code using Hybrid Flow must extract the value `c_hash` from the identification token and compare it with the authorization code.
- `amr` – authentication methods passed, a list of authentication methods passed by the user is indicated. The list may include the following method identifiers:
  - `password` – login using a password;
  - `cls:<method>` (for example, ```cls:password`) – automatic login from a memorized device (in the name of the identifier, after the colon, the authentication method initially passed by the user is indicated, as a result of which the user was memorized on this device);
  - `css` – automatic login based on the results of user registration, password recovery, or switching to a web application from a mobile application using a call using `scope=native`;
  - `sms` – confirmation of login using the code in the SMS message (the second authentication factor);
  - `email` – confirmation of login using the code in the email message (the second authentication factor);
  - `push` – confirmation of login using the code in the push notification to the mobile application (the second authentication factor);
  - `hotp` – login confirmation using a code generated by the HOTP confirmation code generator (the second authentication factor);
  - `totp` – confirmation of login using a code generated by the software TOTP-generator of confirmation codes (the second authentication factor);
  - `tls` – login in automatic authentication mode using TLS Proxy;
  - `spnego` – login using an operating system session;
  - `userApp` – login to the mobile application with a user account linked to the device (Touch ID/Face ID/PIN);
  - `webAuthn` – login using a FIDO2 key (Passkey) or login confirmation using a U2F key;
  - `x509` – login using an electronic signature;
  - `qrCode` – login via QR code;
  - `externalIdps:apple:apple_1` – login using an Apple ID account;
  - `externalIdps:facebook:facebook_1` – login using a Facebook account [Page 302, 1](#);
  - `externalIdps:google:google_1` – login using a Google account;
  - `externalIdps:mail:mail_1` – login using the Mail ID account;
  - `externalIdps:vkid:vkid_1` – login using VK ID account;
  - `externalIdps:ok:ok_1` – login using an account on the Odnoklassniki social network;
  - `externalIdps:vk:vk_1` – login using a VK social network account;
  - `externalIdps:yandex:yandex_1` – login using a Yandex account;

- `outside_methodname` – indicates that the user used an external authentication method named `methodname` during the login process.
- `sid` is the user's session ID;
- additional attributes in accordance with the request to connect the application to Blitz Identity Provider (see possible attributes to include in `id_token` [here](#) (page 298)).

Listing 6: Example of a set of statements

```
{
  "exp": 1445004777,
  "iat": 1444994212,
  "ua_id": "f8a235ff-cb85-4c4b-b55d-544f9358a8d7",
  "sub": "3d10f626-ea77-481d-a50bd4a4d432d86b",
  "amr": [
    "externalIdps:esia:esia_1"
  ],
  "aud": [
    "ais"
  ],
  "iss": "https://login.company.com/blitz",
  "sid": "5a600d12-4b14-447e-ba21-2dc40344a44a"
}
```

**Token signature** It is performed according to the algorithm specified in the `alg` parameter of the token. The signature is calculated from the two previous parts of the token (`HEADER.PAYLOAD`). The Blitz Identity Provider public key certificate required to verify the signature can be downloaded from the following links (located in the `x5c`, attribute, key ID is located in the `kid` attribute):

- <https://login-test.company.com/blitz/.well-known/jwks> (test environment)
- <https://login.company.com/blitz/.well-known/jwks> (production environment)

#### Working with an ID token

1. After receiving the ID token, the application is recommended to validate the ID token, which includes the following checks:
  1. Obtaining the Blitz Identity Provider (`sub`) identifier contained in the ID token and obtaining other additional user attributes required by the application.
  2. Verification of the application identifier, i.e. it is the application that must be specified as the recipient of the ID token.
  3. Verification of the signature of the ID token (using the algorithm specified in the token).
  4. Verification that the current time should be no later than the expiration time of the ID token.

After validating the ID token, the application can consider the user authenticated.

2. To analyze the content of the ID token, as well as to simplify the development of modules for its verification, you can use the available online decoders and libraries.

---

**Tip:** See resources <http://jwt.io/> and [http://kjur.github.io/jsjws/mobile/tool\\_jwt.html#verifier](http://kjur.github.io/jsjws/mobile/tool_jwt.html#verifier).

---

### Checking the access token through the introspection service

The access token data (`access_token`) must be checked in the following cases:

- the application needs to track the expiration date of the token in order to quickly change it to a new one;
- the application has increased security requirements, and the application wants to check the token to make sure that the token is not canceled prematurely. Revocation of the access token (`access_token`) or ID token (`id_token`) may occur for security purposes if the user account password has been reset/changed or if the user account has been blocked;
- the application is a resource provider and provides access to these resources upon presentation of an access token issued by Blitz Identity Provider to the application requesting the resource.

Method `POST https://login.company.com/blitz/oauth/introspect`

**Tip:** See [RFC 7662 OAuth 2.0 Token Introspection](#)<sup>63</sup>.

The introspection service can be called by any system registered in Blitz Identity Provider to verify any access token (the system can verify a token issued to another system). You can check not only the access token, but also the refresh token.

#### Headers

- `Authorization` with the value `Basic {secret}`, where `secret` is `client_id:client_secret` (for example, `app:topsecret`) in Base64 format;
- `Content-Type` with the value `application/x-www-form-urlencoded`.

#### Request body

- `token` is the access token that you want to view the data about.
- Optional parameter `'token_type_hint'` is the type of access token (for example, `"access_token"`), designed to speed up the search.

Returns Access token data:

- `active` is an indication of the validity of the access token, it takes the values `true` or `false`. The token is valid if it was issued by the authorization service Blitz Identity Provider, has not been revoked and its validity has not expired;
- `scope` is the access area to which the access token has been issued. It is transmitted as a list of permissions;
- `client_id` is the identifier of the client system that received this access token;
- `sub` is the identifier of the user (the owner of the resource that provided access to their data), defined as the base identifier in Blitz Identity Provider. The parameter value is returned only if it can be passed within the scope of the presented access token;
- `jti` is the identifier of the access token (in the form of a string);
- `token_type` is the type of the presented access token;
- `iat` is the time when the token was issued (in Unix Epoch);
- `exp` is the expiration time of the token (in Unix Epoch).

<sup>63</sup> <https://tools.ietf.org/html/rfc7662>

## Examples

### Request

```
POST /blitz/oauth/introspect HTTP/1.1
Authorization: Basic cG9ydGFsLmlhc2l1LmxvY2FsOnBvcnRhbC5pYXNpdS5sb2NhbA==
Content-Type: application/x-www-form-urlencoded
```

```
token=MkvRf...No
```

### Response

Listing 7: Valid access\_token

```
{
  "sub": "3d10f626-ea77-481d-a50bd4a4d432d86b",
  "scope": "openid profile",
  "jti": "10jdlNohfHzuv3xoFurvWSPheEJEC7KHdHr-dcaVyYYvV3h012sh",
  "token_type": "Bearer",
  "client_id": "ais",
  "active": true,
  "iat": 1699938503,
  "exp": 1699942103
}
```

Listing 8: Valid id\_token

```
{
  "exp": 1699939472,
  "iat": 1699935872,
  "jti": "fU2FTCzm9G5I4YC6VDFnfjFY5QeIULwH1Yo_BH6OuCQ",
  "token_type": "id_token",
  "active": true,
  "client_id": "ais",
  "sub": "3d10f626-ea77-481d-a50bd4a4d432d86b"
}
```

Listing 9: Valid refresh\_token

```
{
  "sub": "3d10f626-ea77-481d-a50bd4a4d432d86b",
  "scope": "openid profile",
  "jti": "10jdlNohfHzuv3xoFurvWSPheEJEC7KHdHr-dcaVyYYvV3h012sh",
  "token_type": "refresh_token",
  "client_id": "ais",
  "active": true,
  "iat": 1699938503,
  "exp": 1699942103
}
```

Listing 10: Invalid access token

```
{
  "active": false
}
```

### Verification of the access token by the application

When registering an application in Blitz Identity Provider, you can specify that the application should receive an access token (`access_token`) in the `JWT` format. In this case, the application gets the opportunity to independently verify the access token by parsing it.

The structure of the initially received access token will be similar to the structure of *this identification token* (page 309). The secondary access tokens obtained as a result of the exchange of the refresh token (`refresh_token`) will not contain session information (`amr` and additional user attributes will be missing).

Access tokens in the `JWT` format should be used only if the application has special reasons for doing so. In other cases, it is recommended to use regular access tokens in the `opaque` format.

### Logout

If the application provides the user with the opportunity to initiate a logout from the application (logout), then it is not enough for the application to complete a local session to ensure a logout. You must also call the logout operation in Blitz Identity Provider.

If this is not done, then a situation may arise that the user has clicked the button in the application Logout, after which he/she immediately tried to press the button Login, and instead of the expected identification and authentication request, a “Single Sign-On” was triggered, and the user immediately automatically turned out to be logged in.

To initiate a logout in Blitz Identity Provider, after closing its local session, the application must direct the user to Blitz Identity Provider to the URL to perform the logout, passing as parameters:

**Note:** The logout call is performed in accordance with the [OpenID Connect RP-Initiated Logout 1.0 specification](https://openid.net/specs/openid-connect-rpinitiated-1_0.html)<sup>64</sup>.

- Optional parameter ‘`id_token_hint`’ - Blitz Identity Provider checks that the `id_token` of the parameter value is released by it. Valid logout return addresses and logout page design are used according to the configured application with the `client_id` from the `aud` field from the `id_token`.
- Optional parameter ‘`client_id`’ – valid logout return addresses and logout page design are used in accordance with the specified `client_id`.
- Optional parameter ‘`post_logout_redirect_uri`’ is the address of the return to the application after the logout. If the parameter is not set, then redirection to the application after logging out is not performed. If set, it is checked that the value corresponds to at least one allowed return prefix for the application corresponding to the application passed to `id_token_hint` (the `aud` field from `id_token`) or the passed `client_id`. When passing the `post_logout_redirect_uri` parameter, it is also necessary to pass the `id_token_hint` or `client_id` parameter.
- `state` is a set of random characters in the form of a 128-bit request identifier. The same value will be returned in the response when redirecting the user to `post_logout_redirect_uri`.

Example of a logout request:

<sup>64</sup> [https://openid.net/specs/openid-connect-rpinitiated-1\\_0.html](https://openid.net/specs/openid-connect-rpinitiated-1_0.html)

```
https://login.company.com/blitz/oauth/logout?id_token_hint=eyJhbGciOiJSUzI1NiJ9.
↪eyJub...n0=.Ckt...sQ&post_logout_redirect_uri=https://app.company.com/redirect_uri&
↪state=342a2c0c-d9ef-4cd6-b328-b67d9baf6a7f
```

If Blitz Identity Provider completes the logout successfully, it will redirect the user back to the application using the passed URL.

Alternative example of a logout request:

```
https://login.company.com/blitz/oauth/logout?client_id=test-app&post_logout_
↪redirect_uri=https://app.company.com/redirect_uri&state=342a2c0c-d9ef-4cd6-b328-
↪b67d9baf6a7f
```

Valid prefixes of the return pages must be registered in the Blitz Identity Provider settings, otherwise an error will be returned during the logout.

Applications connected to Blitz Identity Provider via OIDC can subscribe to notify them of the user's logout from Blitz Identity Provider. The following features are supported:

- Notification via web browser (Front channel) See [OpenID Connect Front-Channel Logout 1.0<sup>65</sup>](#).
- Notification via the server (Back channel). See [OpenID Connect Back-Channel Logout 1.0<sup>66</sup>](#).

For notification via a web browser, the handler “Link to clear the user's session in the browser (Front channel)” is registered in the application settings in Blitz Identity Provider. If the handler is registered and the user logged into the application during the session, then when the user calls the Blitz Identity Provider logout through the browser on the user's logout page through the frame `<iframe src= "ссылка">`, the application handler specified in the configuration will be called via HTTP GET. If the setting “Add session ID and issuer to the session cleanup link in the browser (Front channel)” was selected, the following parameters will additionally be passed in the request:

- `iss` is the identifier of the identification provider;
- `sid` is the user's session ID.

Example of calling a link to clear a user's session in the browser (Front channel):

```
https://app.company.com/front_channel_logout?iss=https://login.company.com/blitz&
↪sid=4ac78c75-b99d-44dc-9304-d2599c829440
```

In response to the call, the application must terminate the local session and return an *HTTP 200 OK* response. Headers should also be included in the response:

```
Cache-Control: no-cache, no-store
Pragma: no-cache
```

**Note:** When implementing an application-side handler for receiving notifications via a web browser, it is necessary to take into account the features of modern browsers that counteract the transfer of cookies when calling handlers in a frame to URL domains other than the URL domain of the parent page:

- in order for the cookie of a third-party site to be transmitted from the frame, the cookie must have the `SameSite=None` flag and the `Secure` boxes checked, the `X-Frame-Options` header must not be transmitted at the time of setting or overwriting the cookie, and the handler itself must be accessible via HTTPS;
- the handler will not be called in some browsers if the page is opened in Hide ID mode.

For notification via the server, the handler “Link to clear the user's session in the application (Back channel)” is registered in Blitz Identity Provider in the application settings. If the handler is registered and the user logged

<sup>65</sup> [https://openid.net/specs/openid-connect-frontchannel-1\\_0.html](https://openid.net/specs/openid-connect-frontchannel-1_0.html)

<sup>66</sup> [https://openid.net/specs/openid-connect-backchannel-1\\_0.html](https://openid.net/specs/openid-connect-backchannel-1_0.html)



into the application during the session, then when the user calls the logout, the Blitz Identity Provider server will call the application server via HTTP POST to the application handler specified in the configuration. The logout token `logout_token` will be passed to the call, which is a JWT token, the body of which contains the following parameters:

- `iss` is the identifier of the identification provider;
- `aud` – identifiers of notified applications;
- `iat` – the time of the refresh token release;
- `jti` is the ID of the logout token;
- `events` – constant value `http://schemas.openid.net/event/backchannel-logout` according to the OpenID Connect Back-Channel Logout 1.0 specification;
- `sid` is the user's session ID;
- `sub` is the user ID.

The refresh token contains either `sub` (if the setting “Add session ID and issuer to the session clear link in the application (Back channel)” is not enabled) or “`sid`” (if the setting “Add session ID and issuer to the session clear link in the application (Back channel)” is enabled).

Example of calling the user session clear service in the application (Back channel):

```
POST /back_channel_logout HTTP/1.1
Host: app.company.com
Content-Type: application/x-www-form-urlencoded

logout_token=eyJ...J9.eyJ...J9.RV8...Nw
```

Example of the disassembled body of the logout token with the “Add session ID and issuer to the session clear link in the application (Back channel)” setting disabled:

```
{
  "iss": "https://login.company.com/blitz",
  "aud": [
    "ais"
  ],
  "iat": 1646979918,
  "jti": "ee75ccd8-ad30-4175-9a61-3ae06c1a6730",
  "events": {
    "http://schemas.openid.net/event/backchannel-logout": {}
  },
  "sub": "3d10f626-ea77-481d-a50bd4a4d432d86b"
}
```

Example of the disassembled body of the logout token with the “Add session ID and issuer to the session clear link in the application (Back channel)” setting enabled:

```
{
  "iss": "https://login.company.com/blitz",
  "aud": [
    "ais"
  ],
  "iat": 1646979918,
  "jti": "ee75ccd8-ad30-4175-9a61-3ae06c1a6730",
  "events": {
    "http://schemas.openid.net/event/backchannel-logout": {}
  },
  "sid": "4ac78c75-b99d-44dc-9304-d2599c829440"
}
```

In response to a call, the application must:

1. Verify the signature of the logout token by analogy with the verification of the [identification marker signature](#) (page 309).
2. Verify that:
  - `iss` corresponds to the `issuer` of the deployed system;
  - `aud` includes the ID of the called application;
  - the refresh token was released (`iat`) no earlier than 2 minutes ago;
  - `sid` or `sub` correspond to the current user sessions.
3. If any checks of the logout token are unsuccessful, then return the code HTTP 400 Bad Request.
4. If all checks are successful, then terminate the user's local session and return "HTTP 200 OK" if successful or HTTP 501 Not Implemented if the session failed.

It is recommended to include headers in the response:

```
Cache-Control: no-cache, no-store
Pragma: no-cache
```

### 3.2.3 Connecting a mobile app

---

**Tip:** See the [description](#) (page 164) of the interaction between a mobile app and Blitz Identity Provider via OIDC.

---

#### Connection settings

To connect a mobile application to Blitz Identity Provider, you will need the data obtained when [registering it in product](#) (page 295):

- identifier assigned to the application in Blitz Identity Provider (`software_id`);
- initial access token (Initial Access Token);
- application metadata (`software_statement`);
- return URLs registered for the application during authorization;
- logout return URLs registered for the application;
- the permissions registered for the application (`scope`).

In order to interact with Blitz Identity Provider, the application must use the following addresses:

- URL for authorization and authentication:
  - `https://login-test.company.com/blitz/oauth/ae` (test environment)
  - `https://login.company.com/blitz/oauth/ae` (production environment)
- URL for getting and updating the access token:
  - `https://login-test.company.com/blitz/oauth/te` (test environment)
  - `https://login.company.com/blitz/oauth/te` (production environment)
- URL for getting user data:
  - `https://login-test.company.com/blitz/oauth/me` (test environment)
  - `https://login.company.com/blitz/oauth/me` (production environment)
- URL for dynamic registration of a mobile application instance:

- `https://login-test.company.com/blitz/oauth/register` (test environment)
- `https://login.company.com/blitz/oauth/register` (production environment)
- URL for getting access token data:
  - `https://login-test.company.com/blitz/oauth/introspect` (test environment)
  - `https://login.company.com/blitz/oauth/introspect` (production environment)
- URL for performing the logout:
  - `https://login-test.company.com/blitz/oauth/logout` (test environment)
  - `https://login.company.com/blitz/oauth/logout` (production environment)

All these URLs, as well as additional information, are located at the address of dynamically updated settings (metadata) of each Blitz Identity Provider environment:

---

**Tip:** See [RFC 8414 OAuth 2.0 Authorization Server Metadata](https://tools.ietf.org/html/rfc8414)<sup>67</sup>.

---

- `https://login-test.company.com/blitz/.well-known/openid-configuration` (test environment)
- `https://login.company.com/blitz/.well-known/openid-configuration` (productive environment)

Application developers can use a single link to Blitz Identity Provider metadata instead of listing all of the URLs in their application's configuration.

### Ready-made libraries

An information resource <https://appauth.io/>, which provides an SDK for iOS/Android, will be useful for integrating mobile applications with Blitz Identity Provider.

### Dynamic registration of an application instance

Prerequisites for dynamic registration of a mobile application instance:

- the user must install the mobile application;
- the mobile application must have the following data:
  - mobile application ID (`software_id`);
  - initial access token (Initial Access Token);
  - metadata of the mobile application (`software_statement`).

The mobile application must send an HTTP request using the POST method to Blitz Identity Provider to the address of the dynamic registration service `/blitz/oauth/register`.

Parameters must be passed:

- mobile application ID (`software_id`);
- metadata of the mobile application (`software_statement`);
- the type of device on which the mobile application is running (`device_type`) is one of the possible values shown in the table:

---

<sup>67</sup> <https://tools.ietf.org/html/rfc8414>

### Tokens used in Blitz Identity Provider

Device type (device_type)	Description
iphone	Smartphones of the iPhone family
ipad	Tablets of the iPad family
android_phone	Smartphones running Android OS
android_tab	Tablets running Android OS
win_mobile	Devices running Windows 10 Mobile

The dynamic registration request must contain the `Authorization` header with the primary access token (type – Bearer) issued to the application.

Request example:

```
POST /blitz/oauth/register HTTP/1.1
Content-Type: application/json
Authorization: Bearer NINxnizbgYYQg94vEd6MjkTPxR3r2s9IAHBO92AszgTIqItY

{
  "software_id": "CSI",
  "device_type": "iphone",
  "software_statement": "eyJ0e...xQ"
}
```

Upon successful completion of the request, Blitz Identity Provider returns to the instance of the mobile application a list of statements, among which the following are necessary for further work (they must be stored in a secure manner on the user's device):

- ID of the mobile application instance (`client_id`);
- the secret of the mobile application instance (`client_secret`);
- configuration management token (`registration_access_token`);
- configuration management URL (`registration_client_uri`).

Response example:

```
{
  "grant_types": [
    "authorization_code"
  ],
  "registration_client_uri": "https://login.company.com/blitz/oauth/register/dyn~
↪CSI~4e6904c5-ef29-4ae5-8d30-99c359b8270f",
  "scope": "openid profile",
  "registration_access_token": "eyJ0e...tw",
  "client_id": "dyn~CSI~4e6904c5-ef29-4ae5-8d30-99c359b8270f",
  "software_id": "CSI",
  "software_version": "1",
  "token_endpoint_auth_method": "client_secret_basic",
  "response_types": [
    "code"
  ],
  "redirect_uris": [
    "com.example.app:/oauth2redirect/example-provider"
  ],
  "client_secret": "3r0tt20lyeGecWq",
  "client_secret_expires_at": 0
}
```

## User's initial login

After *receiving* (page 318) the `client_id/client_secret` pair, the mobile application instance must identify and authenticate the user according to the OIDC/OAuth 2.0 specifications and taking into account the additional specification [RFC 7636 Proof Key for Code Exchange by OAuth Public Clients](#)<sup>68</sup> (mobile application when interacting with Blitz Identity Provider should use PKCE).

The identification and authentication scenario includes the following steps:

- request for an authorization code;
- getting an access token;
- getting user data in exchange for an access token.

The user's initial login to the mobile application must occur within 1 hour after the completion of dynamic registration in the Blitz Identity Provider instance of the mobile application. Otherwise, the `client_id` will be canceled and a new dynamic registration will be required.

## Getting the authorization code

To authenticate, an instance of the mobile application must call the regular browser of the mobile platform and redirect the user to the URL Blitz Identity Provider of the authorization and authentication service (`/blitz/oauth/ae`).

When using the browser with a mobile application, the following features should be taken into account:

- for iOS, you must use the built-in browser: the `SFSafariViewController` class or the `SFAuthenticationSession` class (in-app browser tab pattern);
- for Android, you need to use the built-in browser: the `Android Custom Tab` function (implements the in-app browser tab pattern).

**Attention:** The use of an Embedded browser is not allowed.

The request parameters should be specified:

- `client_id` is the ID of the mobile application instance;
- `response_type` is a response type (takes the value `code`);
- `scope` is the requested permissions, the `openid` permission must be passed and the necessary additional scope to receive user data (these scope must be provided with metadata);
- `redirect_uri` is a link to return the user to the application, the link must match one of the values specified in the metadata. In order for Blitz Identity Provider to be able to call the mobile application back after authorization, the following schemes should be used:
  - for iOS:

---

**Tip:** For an example of implementation, see: <https://github.com/openid/AppAuth-iOS>

---

\* option 1 is to use the private-use URI scheme (custom URL scheme). Type of return links: `com.example.app:/oauth2redirect/example-provider` (CFBundleURL-Types keys are registered in `Info.plist`);

\* option 2 is to use a URI like "https" (Universal links). Type of return links: `https://app.example.com/oauth2redirect/example-provider` (the "Universal links" function is used, URLs are registered in the `entitlement file` in the application and associated with the application domain). This method is preferable for iOS 9 and above.

---

<sup>68</sup> <https://tools.ietf.org/html/rfc7636>

– for Android:

**Tip:** For an example of implementation, see: <https://github.com/openid/AppAuth-Android>

- \* option 1 is to use the private-use URI scheme (custom URL scheme). Type of return links: `com.example.app:/oauth2redirect/example-provider` (link support using Android Implicit Intents, links are registered in the manifest);
  - \* option 2 is to use a URI like `https` (Universal links). Type of return links: `https://app.example.com/oauth2redirect/example-provider` (available starting from Android 6.0, links are registered in the manifest). This method is preferable for Android 6.0 and higher.
- `state` is a set of random characters in the form of a 128-bit request identifier (used to protect against interception), the same value will be returned in the response – an optional parameter;
  - `access_type` (optional parameter) – whether the application needs to receive `refresh_token`, which is necessary to obtain information about the user in the future when the user is offline. Takes the value “online”/“offline”, `refresh_token` is provided when `access_type=offline`. If the value is not set, then the behavior is determined by the setting set for the specified application in Blitz Identity Provider;
  - `code_challenge_method` is the method for encrypting the request ID, “S256” should be specified;
  - `code_challenge` is the encrypted identifier of the request. The request ID (`code_verifier`) must be stored by the mobile application instance for subsequent transmission to the access token request. The encrypted value is calculated as follows:

```
code_challenge=BASE64URL-ENCODE(SHA256(ASCII(code_verifier)))
```

Example of a request to receive an authorization code (authentication and access token with `openid` and `profile` permissions were requested, PKCE is used):

```
https://login.company.com/blitz/oauth/ae?scope=openid+profile
&access_type=online&response_type=code
&state=342a2c0c-d9ef-4cd6-b328-b67d9baf6a7f
&client_id=dyn~CSI~4e6904c5-ef29-4ae5-8d30-99c359b8270f
&code_challenge_method=S256&code_challenge=qjrzSW9gMiUgpUvqgEPE4
&redirect_uri=https%3A%2F%2Fapp.example.com%2Foauth2redirect%2Fexample-provider
```

An example of a response with the value of the authorization code (`code`) and the `state` parameter:

```
https://app.example.com/oauth2redirect/example-provider?
↪code=f954nEzQ08DXju4wxGbSSfCX7TkZ1GvXUR7TzVus8fGnu4AU1-YIosgax-
↪BLXMeQQAlasD6CN2qG_0KXK5NIjARoKykhuR9IpbuzqeFxS0&state=342a2c0c-d9ef-4cd6-b328-
↪b67d9baf6a7f
```

Possible errors when calling `/oauth/ae` comply with RFC 6749 and are described [here](#)<sup>69</sup>.

<sup>69</sup> <https://tools.ietf.org/html/rfc6749#section-4.1.2.1>

### Getting tokens by an application instance

After receiving the authorization code, the mobile application instance must exchange it for tokens. To do this, the instance must form a POST request to the URL to receive the token. The request must contain the header `Authorization` with the value `Basic {secret}`, where `secret` is “client\_id:client\_secret” (for example, `dyn~CSI~4e69...Wq`) in Base64 format.

Example of a header:

```
Authorization: Basic ZHluOkNTSTo...dx
```

The request body must contain the following parameters:

- `code` is the value of the authorization code that was previously received by an instance of the mobile application from Blitz Identity Provider;
- `grant_type` is the value `authorization_code`;
- `redirect_uri` – must be the same value that was specified in the request to receive the authorization code;
- `code_verifier` is the request ID generated by the mobile application instance when requesting an authorization code.

Request example:

```
POST /blitz/oauth/te HTTP/1.1
Authorization: Basic ZHluOkNTSTo...dx
Content-Type: application/x-www-form-urlencoded
grant_type=authorization_code&code=FLZHS...GU
&redirect_uri=https%3A%2F%2Fapp.example.com%2Foauth2redirect%2Fexample-provider
&code_verifier=M25iVXpKU3puUjFaYWg3T1NDTDQtW1ROUY5YXlwalNoc0hhakxifmZHag
```

An access token and an identification token are returned in response.

Example of a response with successful execution of the request:

```
{
  "id_token": "eyJhb...J9. eyJub...0=.Ckt_dr...sQ",
  "access_token": "d0-xym...BE",
  "expires_in": 3600,
  "scope": "openid profile",
  "token_type": "Bearer"
}
```

After receiving the access token, the instance of the mobile application becomes associated with the user account. It is recommended that the mobile application prompts the user to set a PIN code or enable Touch ID/Face ID.

Besides, using the received access token, the application can [request user data](#) (page 333).

If the authorization code has already been used, the `redirect_uri` did not match the one previously used in the call to `/oauth/ae`, or the code expired, or the `code_verifier` passed does not match `code_challenge`, an error will be returned as a response.

Example of an error response:

```
{
  "error": "invalid_grant",
  "error_description": "The provided authorization grant... is invalid, expired, ↪revoked..."
}
```

Possible errors when calling `/oauth/te` match RFC 6749 and are described [here](#)<sup>70</sup>.

<sup>70</sup> <https://tools.ietf.org/html/rfc6749#section-5.2>

## User re-login

Each time a user logs into an instance of a mobile application, if Internet access is available from the device, the user should be authenticated by calling the Blitz Identity Provider service. In particular, each time you log into an instance of a mobile application, you need to check the user's PIN code or Touch ID/Face ID, then extract the `client_id/client_secret` securely stored on the device and make a request to Blitz Identity Provider to re-log the user. Use the access token received in response from Blitz Identity Provider to get up-to-date user data.

The request to Blitz Identity Provider to re-log in must be made by the POST method to the URL to receive the token (`/oauth/te`). The request must contain the header `Authorization` with the value `Basic {secret}`, where `secret` is the `client_id:client_secret` of the mobile application instance in Base64 format.

The request body must contain parameters:

- `grant_type` is the value `client_credentials`;
- `scope` is a list of permissions requested by the instance of the mobile application.

Request example:

```
POST /blitz/oauth/te HTTP/1.1
Authorization: Basic cG9ydGFsLmlhc2l1LmxvY2FsOnBvcnRhbcC5pYXNpdS5sb2NhbA==
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&scope=profile
```

The response returns an access token and information about this token.

Example of a response with successful execution of the request:

```
{
  "access_token": "dO-xym...BE",
  "expires_in": 3600,
  "scope": "openid profile",
  "token_type": "Bearer"
}
```

Using the received access token, an instance of the mobile application can [request](#) (page 333) up-to-date user data from Blitz Identity Provider in order to visualize or update this data in the device if necessary.

If a user in Blitz Identity Provider revoked the authorization right in Blitz Identity Provider from an instance of a mobile application, the mobile application instance will receive an error as a result of calling Blitz Identity Provider.

Example of an error response:

```
{
  "error": "invalid_client",
  "error_description": "Client authentication failed..."
}
```

Possible errors when calling `/oauth/te` match RFC 6749 and are described [here](#)<sup>71</sup>.

<sup>71</sup> <https://tools.ietf.org/html/rfc6749#section-5.2>



### User switching or logging out

If the mobile application has a user exit or change function, then when the user calls such a function, the mobile application must also call Blitz Identity Provider and delete the `client_id/client_secret` pair released for this instance of the mobile application. If this is not done, then when the user logs out of the mobile application, the user in the web application Blitz Identity Provider *Security Settings* will still see that the mobile application is still linked to his account.

**Note:** The standard address looks like this: `https://login.company.com/blitz/profile`.

To delete the `client_id/client_secret` pair released for an instance of a mobile application from Blitz Identity Provider, the mobile application must send to Blitz Identity Provider a DELETE request to the configuration management URL (`registration_client_uri`) received and stored by the mobile application when *calling the dynamic registration* (page 318) in Blitz Identity Provider instance of the mobile application. The request must contain the header `Authorization` with the value `Bearer {registration_access_token}`, where `registration_access_token` is a configuration management token, also received and stored during the dynamic registration process. The request does not require specifying parameters.

Request example:

```
DELETE /blitz/oauth/register/dyn~CSI~4e6904c5-ef29-4ae5-8d30-99c359b8270f HTTP/1.1
Authorization: Bearer eyJ0e...tw
```

If, after deleting the `client_id/client_secret` pair, the mobile application immediately requests a new `client_id/client_secret` pair and requests user login, then if the previous login was performed in the same browser session, then SSO will work and the user will automatically log in with the previous account. This is usually an undesirable behavior to log in immediately after logging out, since it is expected that the user will want to log in with a different account. Therefore, after logging out, it is recommended to request a new login using one of the following methods:

- When requesting an authorization code, specify the additional parameter `prompt=login` in the request. Then Blitz Identity Provider will prompt the current user to authenticate, even if the Blitz Identity Provider session is active. The user can also select *Change account* on the login page to log in with a different account.
- When requesting an authorization code, specify the additional parameter `prompt=select_account` in the request. So Blitz Identity Provider will immediately prompt the user to select an account from among the remembered ones or log in with a new account. The user does not have to additionally press the button *Change account* on the login page.

### Opening web resources from the application

In some mobile applications, developers may need to provide a function for opening web resources that also require user identification/authentication and use Blitz Identity Provider (end-to-end authentication mode) for this purpose.

When accessing a web resource, a user logged into a mobile application may encounter a situation that Blitz Identity Provider will repeatedly require him to complete identification/authentication in a web resource as a result of a request by the corresponding web application for user identification/authentication in Blitz Identity Provider. To prevent this from happening, the mobile application can immediately request Blitz Identity Provider to receive an access token (`access_token`) for a special permission (`scope`) with the name `native` immediately before calling the web resource.

You can get an access token using the method described in *User re-login* (page 323) or *Getting tokens* (page 304) (if the application has a `refresh_token`).

Request example:

```
POST /blitz/oauth/te HTTP/1.1
Authorization: Basic cG9ydGFsLmlhc2l1LmxvY2FsOnBvcnRhbcC5pYXNpdS5sb2NhbA==
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&scope=native
```

In response, not only the access token and information about this token are returned, but also a special attribute – the end-to-end login marker `css` (cookie short session).

Example of a response with the `css` attribute:

```
{
  "access_token": "d0-xym...BE",
  "css": "nUngQ...LA",
  "expires_in": 3600,
  "scope": "native",
  "token_type": "Bearer"
}
```

After that, the mobile application can open a web resource. At the same time, in the launched web browser, the mobile application must first set a cookie with the following parameters:

- cookie name – `css`;
- cookie domain – `login.company.com`;
- cookie path – `/blitz`;
- flags `HTTPOnly=true` and `Secure=true`;
- the cookie value is the value received in the `css` parameter when receiving an access token from Blitz Identity Provider on the scope named `native`.

If the launched web resource initiates identification (authentication) in Blitz Identity Provider within 300 seconds from the moment of launch, and the cookie was correctly set, then Blitz Identity Provider, at the request of the web application, will automatically perform end-to-end identification and authentication of the user under the account with which the user previously logged into the instance of the mobile application that invoked the web resource.

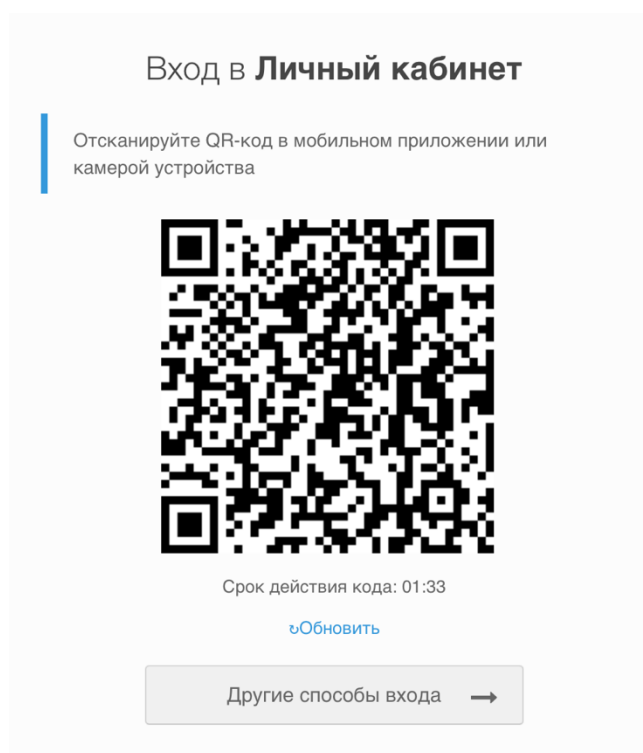
### Login to the application using a QR code

QR code login can be used in Blitz Identity Provider as the first authentication factor (an alternative to entering a username/password). When choosing this login method, Blitz Identity Provider generates and displays to the user a QR code in which the login request is encoded (Figure 6). The validity period of the QR code is limited, and the generated request is a one-time request. Upon expiration of the displayed QR code, the user is given the opportunity to request the display of a new QR code.

The link encoded in the QR code looks like: `QR_URL?code=b0671081-cb73-4839-8bc1-8cf020457228`, for example:

```
https://login.company.com/blitz/login/qr?code=b0671081-cb73-4839-8bc1-8cf020457228
```

The `QR_URL` value can be configured so that if a smartphone is pointed at a QR code using a standard camera application, the user can see a web page with instructions on how to get the correct mobile application to download QR codes or the ability to call a suitable mobile application via Universal Link.



The QR code login process on the mobile application side consists of the following steps:

1. Before photographing the QR code with a mobile application, the user must be logged into the mobile application using Blitz Identity Provider, and the mobile application must receive a valid access token from scope named `blitz_qr_auth` (permission to log in using a QR code) in Blitz Identity Provider.
2. When photographing a QR code, the mobile application should discard the `QR_URL` value (the application does not need it and should be ignored) and the application should read the value of the `code` parameter passed in the link.
3. After reading the QR code, the mobile application should call the Blitz Identity Provider service to receive information about the login request, passing the value of the received code to the service, as well as the header with the access token and the header of the user's current language.

Example of a call:

```
curl --location --request GET 'https://login.company.com/blitz/api/v3/auth/qr/
↪b0671081-cb73-4839-8bc1-8cf020457228' \
--header 'Accept-Language: ru' \
--header 'Authorization: Bearer eyJhb...tA'
```

The response will return a JSON containing information about the IP address, operating system and browser of the device on which the user is trying to log in using a QR code, as well as the name of the application that the user is trying to log in to.

Example of a successful response:

```
{
  "ip": "83.220.238.103",
  "rp_name": "User profile",
  "ip_city": "Москва",
  "browser": "Chrome 109",
  "ip_state": "Москва",
  "os": "macOS 10.15.7",
  "ip_lng": "37.6171",
```

(continues on next page)

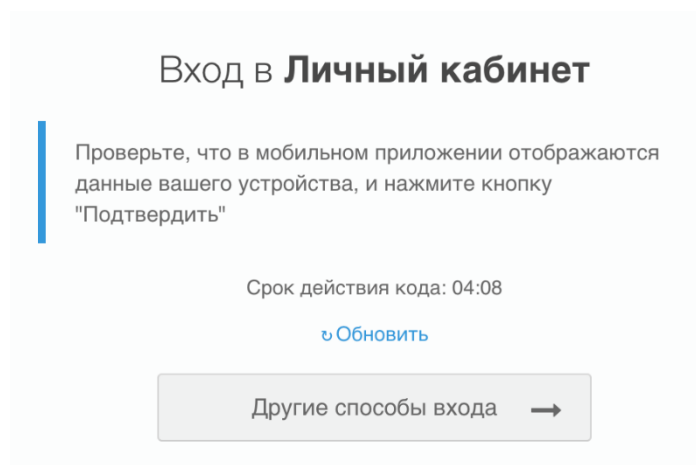
(continued from previous page)

```

"device_type": "pc",
"ip_lat": "55.7483",
"ip_country": "Россия",
"rp_id": "\_blitz_profile",
"device_name": "macOS Big Sur (11)",
"ip_radius": "20",
"device": "PC"
}

```

Besides, the user will be shown a screen in the web page that a login confirmation is expected.



The user in the mobile application needs to display the application name (`rp_name`), IP address (`ip`), geodata (`ip_country`, `ip_state`, `ip_city` - a text description of the address or show on the map at the coordinates `ip_lat`, `ip_lng`), the device used (`device_name`), browser (`browser`).

Possible values of `device_type` now: `kindle`, `mobile`, `tablet`, `iphone`, `windowsPhone`, `pc`, `ipad`, `playStation`, `unknown`. You can use them to visualize a message, or you can simply output the device name as a text string from device.

Example of a response with an invalid access token:

```

{
  "type": "security_error",
  "error": "bad_access_token",
  "desc": "expired_access_token"
}

```

Example of a response with an expired QR code:

```

{
  "type": "process_error",
  "error": "qr_session_expired",
  "desc": "Error while getting QR authentication session"
}

```

Example of a response with an invalid code:

```

{
  "params": {},
  "desc": "Error while getting QR authentication session",
  "error": "qr_session_not_found"
}

```

An example of a response when calling from an already used QR session (when the login has already been confirmed or rejected):

```
{
  "type": "process_error",
  "error": "qr_session_already_completed",
  "desc": "Error while getting QR authentication session"
}
```

1. The mobile application should display the login information received from Blitz Identity Provider JSON to the user, as well as the choice of action: “Allow” or “Reject”. In the case of “Reject”, request the reason for the rejection (“Login caused by error” or “I did not request login”).
2. Depending on the user’s decision, the mobile application should call the Blitz Identity Provider service to confirm or deny login. An access token with scope named `blitz_qr_auth` must be used during the call.

Example of a call when confirming login:

```
curl --location --request POST 'https://login.company.com/blitz/api/v3/auth/qr/
↪5e20b01e-5c7c-4101-8292-98e6865c7bfb/confirm' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer eyJhb...cQ'
```

If successful, HTTP 204 No Content without body will be returned. The user will also log into the application.

If the code is expired, it will be returned:

```
{
  "type": "process_error",
  "error": "qr_session_expired",
  "desc": "Error while confirming QR authentication session"
}
```

If the code does not exist, it will return:

```
{
  "params": {},
  "desc": "Error while confirming QR authentication session",
  "error": "qr_session_not_found"
}
```

Example of a response with an invalid access token:

```
{
  "type": "security_error",
  "error": "bad_access_token",
  "desc": "expired_access_token"
}
```

An example of a response when calling from an already used QR session (when the login has already been confirmed or rejected):

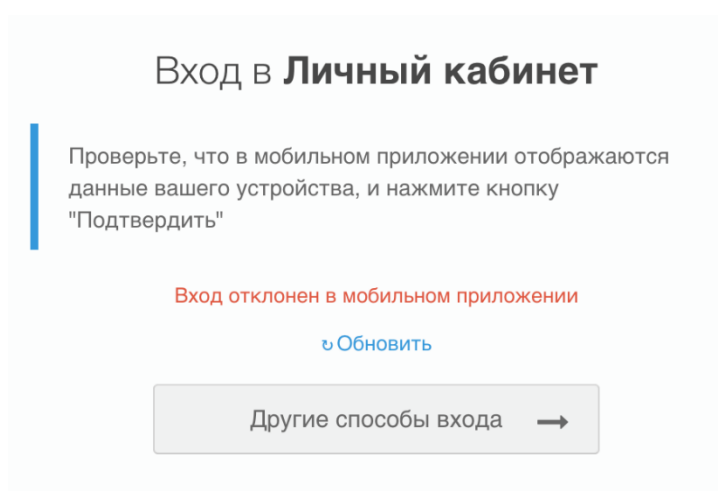
```
{
  "type": "process_error",
  "error": "qr_session_already_completed",
  "desc": "Error while getting QR authentication session"
}
```

An example of a call when login is rejected:

```
curl --location --request POST 'https://login.company.com/blitz/api/v3/auth/qr/
↪845f2334-fa6b-40c0-9a71-f57997166e39/refuse' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer eyJhb...bQ' \
--data-raw '{
"cause_id": "mistake",
"desc": "Вход вызван по ошибке"
}'
```

If login is rejected, you need to pass JSON with the `cause_id` attribute in the request body. It is recommended that if the user rejects the login, ask the reason. If the user reports that he “changed his mind” (or “caused the login by mistake”), then fill in `cause_id=mistake`. But if the user reports that he did not initiate the login, then fill in `cause_id=unauthorized`. The `desc` parameter is optional – you can specify any text string.

If the call is successful, HTTP 204 No Content without body will be returned. The user will also be shown an error screen:



If the code is expired, an error will be returned:

```
{
  "type": "process_error",
  "error": "qr_session_expired",
  "desc": "Error while refusing QR authentication session"
}
```

If the code does not exist, it will return:

```
{
  "params": {},
  "desc": "Error while refusing QR authentication session",
  "error": "qr_session_not_found"
}
```

Example of a response with an invalid access token:

```
{
  "type": "security_error",
  "error": "bad_access_token",
  "desc": "expired_access_token"
}
```

An example of a response when calling from an already used QR session (when the login has already been confirmed or rejected):

```
{
  "type": "process_error",
  "error": "qr_session_already_completed",
  "desc": "Error while getting QR authentication session"
}
```

### 3.2.4 Connecting Smart Device (IoT) applications

#### General information

Blitz Identity Provider supports the ability to authorize smart device applications (voice assistant applications, Smart TV, chatbots) using a user account on another device. For such authorization, the [RFC 8628 OAuth 2.0 Device Authorization Grant](#)<sup>72</sup> is used.

#### Connection settings

In order to interact with Blitz Identity Provider, the application must use the following addresses:

- URL for receiving the authorization confirmation code (OAuth 2.0 Device Authorization Grant):
  - <https://login-test.company.com/blitz/oauth/da> (test environment)
  - <https://login.company.com/blitz/oauth/da> (production environment)
- URL for getting and updating the access token:
  - <https://login-test.company.com/blitz/oauth/te> (test environment)
  - <https://login.company.com/blitz/oauth/te> (production environment)
- URL for getting user data:
  - <https://login-test.company.com/blitz/oauth/me> (test environment)
  - <https://login.company.com/blitz/oauth/me> (production environment)
- URL for getting access token data:
  - <https://login-test.company.com/blitz/oauth/introspect> (test environment)
  - <https://login.company.com/blitz/oauth/introspect> (production environment)
- URL for performing the logout:
  - <https://login-test.company.com/blitz/oauth/logout> (test environment)
  - <https://login.company.com/blitz/oauth/logout> (production environment)

All these URLs, as well as additional information, are located at the address of dynamically updated settings (metadata) of each Blitz Identity Provider environment:

---

**Tip:** See [RFC 8414 OAuth 2.0 Authorization Server Metadata](#)<sup>73</sup>.

---

- <https://login-test.company.com/blitz/.well-known/openid-configuration> (test environment)
- <https://login.company.com/blitz/.well-known/openid-configuration> (production environment)

Application developers can use a single link to Blitz Identity Provider metadata instead of listing all of the URLs in their application's configuration.

<sup>72</sup> <https://www.ietf.org/rfc/rfc8628.html>

<sup>73</sup> <https://tools.ietf.org/html/rfc8414>

## Getting the authorization code

To initiate authorization, the smart device application must make a request to Blitz Identity Provider for the service to receive the authorization confirmation code (`/oauth/da`). The request must be made using the POST method. The request must contain the header `Authorization` with the value `Basic {secret}`, where `secret` is `client_id:client_secret` (for example, `app:topsecret`) in Base64 format.

Example of a header:

```
Authorization: Basic ZHluOkNTSTo...dx
```

The request body must contain the following parameters:

- `client_id` is the application ID;
- `scope` is requested permissions.

Request example:

```
POST /blitz/oauth/da HTTP/1.1
Authorization: Basic ZHluOkNTSTo...dx
Content-Type: application/x-www-form-urlencoded

client_id=test-app&scope=profile
```

In response, Blitz Identity Provider will return the data required to confirm login on another device:

- `device_code` is a device code;
- `user_code` is the authorization request confirmation code displayed to the user;
- `verification_uri` is a link to a page where the user can enter a confirmation code for the authorization request;
- `verification_uri_complete` is a link to a page where the authorization request confirmation code has already been substituted as a parameter;
- `expires_in` is the lifetime of the user code in seconds;
- `interval` is the recommended waiting period in seconds when the application asks the user to enter the authorization request confirmation code.

Example of a response with successful execution of the request:

```
{
  "device_code": "7Lz30lK57bWaKHBxM8kW7KpOFvDg_4ujz3LpQxcleE",
  "user_code": "934-367-578",
  "verification_uri": "https://device.company.com",
  "verification_uri_complete": "https://device.company.com?uc=934-367-578",
  "expires_in": 300,
  "interval": 5
}
```

Upon receiving a response, the smart device application should instruct the user to click on the `verification_uri` link and enter the code from `user_code`.

**Note:** The link in `verification_uri` is displayed according to the settings set in Blitz Identity Provider. It is recommended to configure this link to be short and easy for users to enter, as well as well perceived by ear or beautifully displayed on the Smart TV screen. From this link, redirection should be configured to the handler for user input of the confirmation code located on the page `https://login.company.com/blitz/oauth/device?ci=client_id`, where instead of `client_id` you need to set the ID of the application registered in Blitz Identity Provider, from the settings of which the allowed login methods and settings for the appearance of the login page will be taken.



Depending on the type of smart device, you need to choose the most user-friendly method. For example:

- When logging in to a Smart TV, the application can draw the user a QR code in which encode the link from `verification_uri_complete`. Then the user will need to point the phone's camera at the QR code and log in on the phone.
- When logging in to the chatbot, the application can draw the user a button that opens a link from `verification_uri_complete`. in the browser.' Then the user will need to log in to their device's browser.
- When logging in to the voice assistant application, the application can instruct the user which site he should go to and voice the code that the user must enter, or the application can send the user an SMS message or an e-mail with instructions.

### Getting a security token

After providing instructions to the user, the smart device application should start polling Blitz Identity Provider with an interval from the `interval` parameter to obtain security tokens. To do this, the application must access Blitz Identity Provider using the POST method at the URL to receive a token (`/oauth/te`). The request must contain the header `Authorization` with the value `Basic {secret}`, where `secret` is the `client_id:client_secret` of the mobile application instance in Base64 format.

The request body must contain parameters:

- `grant_type` is the value `urn:ietf:params:oauth:grant-type:device_code`;
- `device_code` is the previously received device code.

Request example:

```
POST /blitz/oauth/te HTTP/1.1
Authorization: Basic cG9...A==
Content-Type: application/x-www-form-urlencoded

grant_type=urn:ietf:params:oauth:grant-type:device_code&device_code=Yrn...\_0
```

If the user has not yet confirmed authorization, Blitz Identity Provider will return the following response with an error:

```
{
  "error": "authorization_pending",
  "error_description": "The authorization request is still pending"
}
```

If the user code has expired or the code is incorrect, Blitz Identity Provider will return the following error response:

```
{
  "error": "invalid_grant",
  "error_description": "The provided authorization grant (e.g., authorization_
↪code, resource owner credentials) or refresh token is invalid, expired, revoked,
↪does not match the redirection URI used in the authorization request, or was_
↪issued to another client."
}
```

If the user has confirmed authorization, Blitz Identity Provider will return the access token and information about it to the application, as well as the update token.

Example of a response with successful execution of the request:

```
{
  "access_token": "eyJ...tA",
```

(continues on next page)

(continued from previous page)

```

"refresh_token": "wVE...cw",
"scope": "profile",
"token_type": "Bearer",
"expires_in": 3600
}

```

Using the received access token, the smart device application can *запросить* (page 333) up-to-date user data from Blitz Identity Provider.

### 3.2.5 Getting user attributes

To request user data, you must make a request using the GET method at the URL of receiving user data (`/oauth/me`). The following header should be added to the request:

```
Authorization: Bearer <access token>
```

In the header, `<access token>` is the access token received from Blitz Identity Provider (see [Getting tokens](#) (page 304) and [Getting tokens by an application instance](#) (page 322)).

Request example:

```

GET /blitz/oauth/me HTTP/1.1
Authorization: Bearer NINxn...tY
Cache-Control: no-cache

```

The response will display only the data that *are defined in the scope* (page 298) to which the access token was received.

Response example:

```

{
  "family_name": "Иванов",
  "given_name": "Иван",
  "middle_name": "Иванович",
  "email": "iivanov@company.com",
  "phone_number": "79162628910",
  "sub": "3d10f626-ea77-481d-a50bd4a4d432d86b"
}

```

A user account can be included in user groups. To get a list of groups that a user is included in, an access token must be obtained from scope named `usr_grps`.

An example of a response for a user included in access groups:

```

{
  "family_name": "Иванов",
  "given_name": "Иван",
  "middle_name": "Иванович",
  "email": "iivanov@company.com",
  "phone_number": "79162628910",
  "sub": "3d10f626-ea77-481d-a50bd4a4d432d86b",
  "groups": [
    {
      "id": "564486ff-af0a-3fb1-3f09-e7c5f7f9833e",
      "name": "Тестовая организация",
      "OGRN": "1234567890123",
      "INN": "9876543210"
    }
  ]
}

```

(continues on next page)

(continued from previous page)

```
]
}
```

### 3.2.6 Ensuring connection security

The operator of the application connected to Blitz Identity Provider must ensure compliance with the following security requirements:

1. The confidentiality of the `client_secret` value received for the application during registration in Blitz Identity Provider must be ensured:
  - It is forbidden to betray the value of `client_secret` to persons who are not related to the operation of the application.
  - It is forbidden to use `client_secret` in the client part of the software (code executed on the side of the browser, mobile application, desktop application). `client_secret` should be used only in the server components of the application. The exception is the `client_secret` received by a mobile or desktop application using a dynamic registration operation, such a `client_secret` can be stored and processed in a mobile or desktop application.
  - If the `client_secret` is compromised, then an application must be submitted to replace the `client_secret` application. Blitz Identity Provider allows for “smooth replacement” of `client_secret`, namely, an additional `client_secret` can be assigned to the application for the time while the application is being reconfigured from the old to the new value `client_secret`.
2. The confidentiality of access tokens (`access_token`) and refresh tokens (`refresh_token`) received by the application from Blitz Identity Provider must be ensured.
  - You should avoid using access tokens in the browser part of the application. If it is still necessary (SPA application), then the JS code using the access token should provide protection against the possibility of obtaining the value of the access token from the browser console.
  - It is forbidden to store/process the update token on the side of the browser part of the application – the update token must be used exclusively in the server components of the application. When storing update tokens in an application (in databases, files, etc.), access to stored update tokens must be limited.
3. The application’s interaction with Blitz Identity Provider in the production loop should be carried out exclusively using a secure connection (HTTPS). It is forbidden to use HTTP in application handlers (return addresses `redirect_uri`, `post_logout_redirect_uri`).
4. The application is not allowed to open the Blitz Identity Provider login page in the frame.
5. When connecting mobile applications to Blitz Identity Provider:
  - using PKCE is mandatory;
  - it is forbidden to use an Embedded browser.

## 3.3 SAML application integration

### 3.3.1 How to register the application correctly

Authentication in SAML terminology is the result of the interaction of three parties:

- the identity provider (`Identity Provider`), which is Blitz Identity Provider;
- the service provider (`Service Provider`), which is the connected application;
- the user’s web browser (`User Agent`).

The first step when connecting an application is to [register](#) (page 171) it as a service provider in Blitz Identity Provider. You must first prepare an XML file with the metadata of the service provider or the parameter values necessary for self-preparation of metadata.

The metadata of the service provider describes the settings for connecting the application to Blitz Identity Provider (for example, the URL of the application endpoints, keys for checking the item instance). The XML language is used to describe metadata.

---

**Tip:** [See more about SAML metadata](#)<sup>74</sup>.

---

**Attention:** Metadata should be prepared based on the results of the work performed for [adding the protocol support](#) (page 336).

If the application is a ready-made software that supports SAML, then the metadata must be obtained according to the documentation for this software. Usually, such software provides a URL where metadata can be obtained.

If the software of the connected application does not provide for downloading metadata, but the software documentation describes the parameters that must be configured to connect the application, then you can specify these parameters so that the metadata based on them is independently prepared by the Blitz Identity Provider Administrator.

In this case, you must specify the following parameters:

1. Service Provider ID (`entityID`) – should be specified only if the application needs a specific `entityID`. Otherwise, the `entityID` will be independently assigned by the Blitz Identity Provider Administrator.
2. Application (service provider) public key certificate – should be specified only if the application signs the SAML request when sending to Blitz Identity Provider.

**Note:** The service provider certificate is different from the TLS certificate of the connected website. This is usually a self-signed certificate with a long validity period.

---

**Important:** RSA-2048 keys must be used.

---

**Note:** It is acceptable to use self-signed certificates with a long validity period.

3. URL for receiving a response from Blitz Identity Provider SAML - the application must provide a handler that receives SAML-responses from Blitz Identity Provider with login results. This application setup is usually called `Assertion Consumer Service`.
4. The URL for receiving a logout request from Blitz Identity Provider is a selective setting. If the application supports a single logout, then it can provide a single logout handler. This application setting is usually called `Single Logout Service Location`.
5. The URL for redirecting the user to the application after a successful logout is an optional setting. If the application supports a single logout and can initiate a single logout, then it can provide a URL to return the user after the logout. This application setting is usually called `Single Logout Service Response Location`.
6. The list of requested attributes (`SAML Assertion`).

---

<sup>74</sup> <https://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf>

### Available user attributes

Attribute	Description
logonname	Username of the user in the domain
surname	Last name
firstname	Name
middlename	Patronymic
email	Business email address

7. Indicates whether attributes must be transmitted in encrypted form.

**Note:** Attributes in a SAML message are always passed signed. It is advisable to enable attribute encryption if the user should not be able to read the attribute value.

### 3.3.2 Connecting the application via SAML

#### Connection data

To connect an application to Blitz Identity Provider, you will need the data obtained during its [registration](#) (page 334):

- the identifier assigned to the application in Blitz Identity Provider (`entityID`);
- the metadata file of the service provider.

The application interacts with Blitz Identity Provider services using the following addresses:

- Blitz Identity Provider metadata:
  - `https://login-test.company.com/blitz/saml/profile/Metadata/SAML` (test environment)
  - `https://login.company.com/blitz/saml/profile/Metadata/SAML` (production environment)
- URL for authentication:
  - `https://login-test.company.com/blitz/saml/profile/SAML2/Redirect/SSO` (test environment)
  - `https://login.company.com/blitz/saml/profile/SAML2/Redirect/SSO` (production environment)
- URL for the logout:
  - `https://login-test.company.com/blitz/saml/profile/SAML2/Redirect/SLO` (test environment)
  - `https://login.company.com/blitz/saml/profile/SAML2/Redirect/SLO` (production environment)
- Publisher's URL:
  - `https://login-test.company.com/blitz/saml/` (test environment)
  - `https://login.company.com/blitz/saml/` (production environment)

If the application supports the SAML connection protocol, then the specified data should be sufficient to configure the application. If the application does not support the SAML protocol, it should be modified according to the recommendations set out in the sections: [Ready-made libraries](#) (page 338) and [Principle of integration](#) (page 339).

Typical questions about how to set up an application to connect to Blitz Identity Provider over the SAML protocol:

### Where can I find the metadata of the identity provider?

To download the metadata, follow the link <https://login.company.com/blitz/saml/profile/Metadata/SAML> and copy the open XML document into the application.

### Where can I find the SAML certificate of the identity provider?

Open the XML document with the metadata of the identity provider. Find the section `<ds:X509Certificate></ds:X509Certificate>` – this is where the SAML certificate of the identification provider is located. Example:

```

▼<EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:shibmd="urn:mace:shibboleth:metadata:1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" entityID="https://sudir.mos.ru/blitz/saml">
  ▼<IDPSSODescriptor protocolSupportEnumeration="urn:mace:shibboleth:1.0
    urn:oasis:names:tc:SAML:1.1:protocol urn:oasis:names:tc:SAML:2.0:protocol">
    ▼<Extensions>
      <shibmd:Scope regexp="false">0.1</shibmd:Scope>
    </Extensions>
    ▼<KeyDescriptor>
      ▼<ds:KeyInfo>
        ▼<ds:X509Data>
          ▼<ds:X509Certificate>
            MIIDDzCCAFegAwIBAgIJANjxtiKgDpaeMA0GCSqGSIb3DQEBBQUAMBcxFTATBgNV
            BAMTDHh1ZGlyLm1vcy5ydTAeFw0xODA2MjAxNjQ2MDZaFw0yODA2MTcxNjQ2MDZa
            MBcxFTATBgNVBAMTDHh1ZGlyLm1vcy5ydTCCASIwDQYJKoZIhvcNAQEBBQADggEP
            ADCCAQoCggEBANK5Ue/3dmNTLdTzKNrgKLM71pdnBFNjNjDkKkBF2GodQ+r+ePLz
            thw5Gn9G4uLmwFol13fU6usbEdi2IDzg3M5s1T8YbCcxvaw7ddNU9Jdh1YAQIrXT
            VvtRCajyZk3AwraXNj1Ai9Qq8XuXS1EtlymvdUAeY1SScKDPNYIM8cqdHmvSXXVx
            FggJn+S1l6MEDv/0quM2MvOhgLuP7i6J8wNXD4P4fz8+oNGPcQLwn90fIGgFyPBE
            nQ2vmEn0NRotwQCnYcIAPeQ9jMBGIMi2yQtIsjFYDjddqBqau/cXuVybiYA8om3W
            cyMIDFdcJ2RAAHtzNdXN8xnnv8IMrqrqG/MCAwEAANeMFwwOwYDVR0RBDAQwMoIw
            c3VkaXIubW9zLnJ1O1VSSUpodHRwczovL3N1ZGlyLm1vcy5ydS9ibG10ei9zYW1s
            MB0GA1UdDgQWB8Rw3ACqmoCP31aMlW/KtwFsQLZ7iDANBgkqhkiG9w0BAQUFAAOCC
            AQEAJ72xDGx37QBdHIYDiOhwe1Kxibvwm5DZxQ6Sc6YT56fnCwdJeUllJ82yK0Iiw
            Hwfnre+nRRuAHLA9DhaZiYmBvUuqE1tBYadwqIKS01518khE509jnmYizWMiwRPK
            IUz730BQUd13zsT+Wv021Xced8PKR73Y2XZCnIyDbYNIpy1ST9V0/bkB1S6VR8x
            00iOr89rgY/1EwXRnQn+9Wm2tQZXdCTHOBg7kCg4M4OnqyO1irfUvoHboeVrLUA
            ap/b+fHRdl2p08qCJOSCRhPwE TuyYolqt3DSYJqqTDuilTyg8i61j65xL01JER9J
            48L3KzS5SY/DUHYmflfddIRb/Q==
          </ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
    </KeyDescriptor>
    <ArtifactResolutionService Binding="urn:oasis:names:tc:SAML:1.0:bindings:SOAP-binding"
      Location="https://sudir.mos.ru/blitz/saml/profile/SAML1/SOAP/ArtifactResolution" index="1"/>
    <ArtifactResolutionService Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
      Location="https://sudir.mos.ru/blitz/saml/profile/SAML2/SOAP/ArtifactResolution" index="2"/>
    <SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
      Location="https://sudir.mos.ru/blitz/saml/profile/SAML2/Redirect/SLO"
      ResponseLocation="https://sudir.mos.ru/saml/profile/SAML2/Redirect/SLO"/>
    <SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Plain-Redirect"
      Location="https://sudir.mos.ru/blitz/saml/profile/SAML2/Redirect/Plain/SLO"
      ResponseLocation="https://sudir.mos.ru/saml/profile/SAML2/Redirect/Plain/SLO"/>
    <SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
      Location="https://sudir.mos.ru/blitz/saml/profile/SAML2/SOAP/SLO"/>
    <NameIDFormat>urn:mace:shibboleth:1.0:nameIdentifier</NameIDFormat>
  </IDPSSODescriptor>
</EntityDescriptor>

```

Sometimes, in order to load correctly into the application, you need to insert the line -----BEGIN CERTIFICATE----- before the line with the certificate, and after -----END CERTIFICATE-----

### Where can I find the addresses of the SAML handlers of the identity provider?

The application should send identification/authentication requests to the following handlers (SingleSignOnService) in the PROD-environment:

- <https://login.company.com/blitz/saml/profile/SAML2/Redirect/SSO> – a standard SAML handler is used to receive requests compressed using the Deflate algorithm.
- <https://login.company.com/blitz/saml/profile/SAML2/Redirect/Plain/SSO> – for receiving uncompressed requests – should be used only if the connected application does not use deflate.

The application should send requests for a single logout to the following handlers (SingleLogoutService) in the PROD-environment:

- <https://login.company.com/blitz/saml/profile/SAML2/Redirect/SLO> – a standard SAML handler is used to receive requests compressed using the Deflate algorithm.
- <https://login.company.com/blitz/saml/profile/SAML2/Redirect/Plain/SLO> – for receiving uncompressed requests – should be used only if the connected application does not use deflate.

In the TEST environment, similar addresses start with <https://login-test.company.com>.

### What is the entity ID of the identity provider?

Blitz Identity Provider as an identification provider, it has the following entityID:

- For the PROD-environment – <https://login.company.com/blitz/saml>
- For the TEST-environment – <https://login-test.company.com/blitz/saml>

### Ready-made libraries

Since self-development of the SAML client software interface is a time-consuming task, and implementation errors are fraught with security threats, it is recommended to use existing popular SAML client libraries when integrating an application using SAML:

- [OIOSAML<sup>75</sup>](#) (Java, .NET),
- [OpenSAML<sup>76</sup>](#) (Java),
- [Spring Security SAML<sup>77</sup>](#) (Java),
- [SimpleSAMLphp<sup>78</sup>](#) (PHP),
- [ruby-saml<sup>79</sup>](#) (Ruby on Rails).

The following are the key information needed to understand the SAML authentication process.

<sup>75</sup> <https://digitaliser.dk/group/42063/resources>

<sup>76</sup> <https://wiki.shibboleth.net/confluence/display/OS30/Home>

<sup>77</sup> <https://spring.io/projects/spring-security-saml>

<sup>78</sup> <https://simplesamlphp.org/>

<sup>79</sup> <https://rubygems.org/gems/ruby-saml/>



## Principle of integration

To connect to Blitz Identity Provider in order to identify and authenticate users, the application can use the [SAML standard](#)<sup>80</sup> versions 1.0, 1.1, 2.0.

In this case, the process of interaction between the application and Blitz Identity Provider should be built in accordance with the profile [SAML Web Browser SSO Profile](#)<sup>81</sup>.

The SAML standard is based on XML and defines ways to exchange information about user authentication and their identification data (attributes, permissions).

In order to be able to interact, the service provider and the identity provider must first exchange interaction settings described in the form of XML documents and called metadata. The service provider should receive the Blitz Identity Provider settings called [identity provider metadata](#) (page 334).

## Identification and authentication

See the [description](#) (page 166) of the interaction between a web app and Blitz Identity Provider via SAML.

## Logout

An application connected to Blitz Identity Provider by SAML may also provide for the possibility of implementing a single logout. For these purposes Blitz Identity Provider supports [SAML Single Logout Profile](#)<sup>82</sup>. The application can send a `<LogoutRequest>` SAML-request to Blitz Identity Provider and, if the single logout is completed successfully, receive a `<LogoutResponse>` SAML-response from Blitz Identity Provider. If the application should be involved in a single logout initiated by another application connected to Blitz Identity Provider, then it should also provide the ability to process `<LogoutRequest>` requests received by the application from Blitz Identity Provider. In case of successful completion of the local session, the application should notify Blitz Identity Provider by sending it a SAML response `<LogoutResponse>`.

## 3.4 User management API

### 3.4.1 General information

#### REST API versions

Currently, the following versions of the REST API are available in Blitz Identity Provider, which differ in the authorization method:

**Warning:** Services of versions `v1` and `v2` after the appearance of analogues in the newer `v3` will be marked as obsolete, and it will be recommended to switch from their use to services `v3`.

- `v1` – REST services available at the following addresses:
  - `https://login.company.com/blitz/reg/api/v1/`,
  - `https://login.company.com/blitz/api/v1/`.

HTTP Basic authorization is used to authorize calls to these services. For an application that will call REST services, you must set a password in the application settings on the REST tab of the application protocol settings. All `v1` REST services will be available to the application.

<sup>80</sup> <http://saml.xml.org/saml-specifications>

<sup>81</sup> <https://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf>

<sup>82</sup> <https://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf>



---

**Tip:** If you do not plan to use any of the services, disable their call through the web server settings (nginx).

---

- v2 – REST services available at `https://login.company.com/blitz/api/v2/`. HTTP Basic authorization is used to authorize calls to most of these services, and OAuth 2.0 is used for some services.
- v3 – REST services available at `https://login.company.com/blitz/api/v3/`. Oath 2.0 and security tokens received from Blitz Identity Provider are used to authorize calls to these services. Applications' access to various REST services is regulated through permissions (`scope`).

### REST API access modes

Provided by Blitz Identity Provider services `https://login.company.com/blitz/api/v3/` can be called in two modes:

- user mode,
- system mode.

### User access mode

In user mode, the service is called with rights in relation to the account of the currently authorized user. When calling the service, the following headers must be passed:

- **Authorization:** `Bearer <access token with user permissions>` – authorization header containing an access token with *permissions of the* (page 340) of the current user.
- **X-Forwarded-For:** `<user IP address>` is the header in which the value of the user's IP address should be transmitted. This value will be recorded in the security event Blitz Identity Provider.
- **User-Agent:** `<User-Agent value>` is the header in which the value `User-Agent` of the user's device should be passed. This value will be recorded in the Blitz Identity Provider security event.

### Possible user permissions

#### Changing the password

`blitz_change_password`

To use the POST `/blitz/api/v2/users/{subjectId}/password` service.

#### Account rights management

`blitz_user_rights`

To use the services:

- GET `/blitz/api/v3/rights/of/{subjectId}`,
- POST `/blitz/api/v2/users/rights/change`.

### Getting attributes

blitz\_api\_user

To use the GET `/blitz/api/v3/users/{subjectId}` service.

### Changing attributes

blitz\_api\_user\_chg

To use the POST `/blitz/api/v3/users/{instanceId}` service.

### Getting two-factor authentication settings, permissions, security question

blitz\_api\_usec

To use the services:

- GET `/blitz/api/v3/users/{subjectId}/auth`,
- GET `/blitz/api/v3/users/{subjectId}/totps`,
- GET `/blitz/api/v3/users/{subjectId}/acls`,
- GET `/blitz/api/v3/users/{subjectId}/secQsn`,
- POST `/blitz/api/v3/users/{subjectId}/secQsn /check`.

### Changing the password, resetting sessions, changing the security question, two-factor authentication settings, revoking permissions

blitz\_api\_usec\_chg

To use the services:

- POST `/blitz/api/v3/users/{instanceId}/pswd`,
- POST `/blitz/api/v3/users/{instanceId}/sessions/reset`,
- POST `/blitz/api/v3/users/{instanceId}/secQsn`,
- POST `/blitz/api/v3/users/{subjectId}/auth`,
- GET `/blitz/api/v3/users/{subjectId}/totps /attach/qr`,
- POST `/blitz/api/v3/users/{subjectId} /totps /attach/qr`,
- DELETE `/blitz/api/v3/users/{subjectId} /secQsn`,
- DELETE `/blitz/api/v3/users/{subjectId} /totps/{id}`,
- DELETE `/blitz/api/v3/users/{subjectId} /acls/{id}`.

### Getting memorized devices

blitz\_api\_uapps

To use the GET `/blitz/api/v3/users/{subjectId}/apps` service.

### Deleting memorized devices

blitz\_api\_uapps\_chg

To use the DELETE `/blitz/api/v3/users/{subjectId}/apps/{id}` service.

### Getting security events

blitz\_api\_uaud

To use the GET `/blitz/api/v3/users/{subjectId}/audit` service.

### Getting a list of external provider accounts

blitz\_api\_ufa

To use the GET `/blitz/api/v3/users/{subjectId}/fa` service.

### Changing the list of external provider accounts

blitz\_api\_ufa\_chg

To use the services:

- POST `/blitz/api/v3/users/{subjectId}/fa/{fpType}/{fpName}/{sid}`,
- DELETE `/blitz/api/v3/users/{subjectId}/fa/{fpType}/{fpName}/{sid}`.

### Login using a QR code

blitz\_qr\_auth

To use the services:

- GET `/blitz/api/v3/auth/qr/{QR_code}`,
- POST `/blitz/api/v3/auth/qr/{QR_code}/confirm`,
- POST `/blitz/api/v3/auth/qr/{QR_code}/refuse`.

The application receives an access token for user permissions at the time of user identification and authentication.

**Note:** The identification and authentication mechanisms are described in the sections:

- [Getting the authorization code](#) (page 300)
- [Getting tokens](#) (page 304)

### System access mode

This section provides a list of permissions that an application can get to access the REST API.

### Possible system permissions (permissions granted to the application)

#### Access to services for working with organizations

blitz\_groups

To use the services:

- GET /blitz/api/v2/grps/{id},
- POST /blitz/api/v2/grps,
- POST /blitz/api/v2/grps/{id}?profile={profile},
- DELETE /blitz/api/v2/grps/{id}?profile={profile},
- GET /blitz/api/v2/grps/{id}/members,
- POST /blitz/api/v2/grps/{id}/members/add?profile={profile},
- POST /blitz/api/v2/grps/{id}/members/rm?profile={profile}.

#### Assigning and revoking access rights

blitz\_rights\_full\_access

To use the services:

- PUT /blitz/api/v3/rights,
- DELETE /blitz/api/v3/rights,
- GET /blitz/api/v3/rights/on,
- GET /blitz/api/v3/rights/of.

#### Revocation of access rights for slave accounts

blitz\_rm\_rights

To use the POST /blitz/api/v2/users/rights/change service.

#### Getting attributes of any user

blitz\_api\_sys\_users

To use the GET /blitz/api/v3/users/{subjectId} service.

#### Changing attributes of any user

blitz\_api\_sys\_users\_chg

To use the POST /blitz/api/v3/users/{instanceId} service.

### Registration of user account

blitz\_api\_sys\_users\_reg

To use the PUT `/blitz/api/v3/users` service.

### Getting two-factor authentication settings, permissions of any user, security question

blitz\_api\_sys\_usec

To use the services:

- GET `/blitz/api/v3/users/{subjectId}/auth`,
- GET `/blitz/api/v3/users/{subjectId}/totps`,
- GET `/blitz/api/v3/users/{subjectId}/acls`,
- GET `/blitz/api/v3/users/{subjectId}/state`,
- GET `/blitz/api/v3/users/{subjectId}/secQsn`,
- POST `/blitz/api/v3/users/{subjectId}/secQsn/check`.

### Changing the password, two-factor authentication settings and security question, resetting sessions, revoking permissions of any user

blitz\_api\_sys\_usec\_chg

To use the services:

- POST `/blitz/api/v3/users/{instanceId}/pswd`,
- POST `/blitz/api/v3/users/{instanceId}/sessions/reset`,
- POST `/blitz/api/v3/users/{subjectId}/auth`,
- POST `/blitz/api/v3/users/{subjectId}/state`,
- GET `/blitz/api/v3/users/{subjectId}/totps/attach/qr`,
- POST `/blitz/api/v3/users/{subjectId}/totps/attach/qr`,
- POST `/blitz/api/v3/users/{subjectId}/secQsn`,
- DELETE `/blitz/api/v3/users/{subjectId}/totps/{id}`,
- DELETE `/blitz/api/v3/users/{subjectId}/acls/{id}`,
- DELETE `/blitz/api/v3/users/{subjectId}/secQsn`.

### Getting any user's devices

blitz\_api\_sys\_uapps

To use the GET `/blitz/api/v3/users/{subjectId}/apps` service.

### Deleting any user's devices

blitz\_api\_sys\_uapps\_chg

To use the DELETE /blitz/api/v3/users/{subjectId}/apps/{id} service.

### Getting security events for any user

blitz\_api\_sys\_uaud

To use the GET /blitz/api/v3/users/{subjectId}/audit service.

### Getting a list of external provider accounts

blitz\_api\_sys\_ufa

To use the POST /blitz/api/v3/users/{subjectId}/fa/{fpType}/{fpName}/{sid} service.

### Changing the list of external provider accounts

blitz\_api\_sys\_ufa\_chg

To use the DELETE blitz/api/v3/users/{subjectId}/fa/{fpType}/{fpName}/{sid} service.

### Obtaining an access token issued by any external identity provider

fed\_tkn\_any

You can [configure](#) (page 109) Blitz Identity Provider to store user access tokens issued by external identity providers. This permission allows you to retrieve a stored access token issued by any provider.

To use the GET /blitz/api/v3/users/\${subjectId}/fedToken/\${fedPointType}/\${fedPointName} service.

### Obtaining an access token issued by a specific external provider

fed\_tkn\_\${fedPointType}\_\${fedPointName}

You can [configure](#) (page 109) Blitz Identity Provider to store user access tokens issued by external identity providers. This permission allows you to retrieve a stored access token issued by a provider with the \${fedPointType} type and \${fedPointName} name.

To use the GET /blitz/api/v3/users/\${subjectId}/fedToken/\${fedPointType}/\${fedPointName} service.

To get an access token for system permission, the application must make a request to get a token:

- Request POST https://login.company.com/blitz/oauth/te.
- The request must contain the header Authorization with the value "Basic {secret}", where secret is client\_id:client\_secret (for example, app:topsecret) in Base64 format.
- The request body must contain the following parameters:
  - grant\_type - takes the value client\_credentials;
  - scope is the requested system permission.

- In response, the application will receive an access token `access_token`, its lifetime `expires_in` and the token type `token_type`.

---

**Tip:** It is recommended that the application caches the received access token for repeated use for a time slightly less than the `expires_in` parameter, after which it receives a new access token for updating in the cache.

---

- Possible errors when calling `/oauth/te` match RFC 6749 and are described [here](#)<sup>83</sup>.

## Examples

### Header

```
Authorization: Basic YWlzOm...XQ=
```

### Request

```
POST blitz/oauth/te HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Authorization: Basic ZG5ld...lg

grant_type=client_credentials&scope=blitz_groups
```

### Response

```
{
  "access_token": "QFiJ9mPgERPusd36mQvD4mfzYolH_CmuddAJ3YKTOI",
  "expires_in": 3600,
  "scope": "blitz_groups",
  "token_type": "Bearer"
}
```

### Error

When trying to call a REST service with an expired access token to it: HTTP 401 Unauthorized.

## 3.4.2 Accounts

This section contains the REST API for managing user accounts.

---

<sup>83</sup> <https://tools.ietf.org/html/rfc6749#section-5.2>

## Registration

Method PUT `https://login.company.com/blitz/reg/api/v3/users`

Registration of a user account.

Required permissions: `blitz_api_sys_users_reg`.

Headers To send an e-mail in English, specify the `Accept-Language: en` header (available only in v3).

Request body

### user.attrs block

Attributes of the account being registered:

- `first_name` is a surname;
- `name` is the name;
- `middle_name` is a middle name;
- `phone_number` is a mobile phone number in the form of a composite object with attributes:
  - `value` is a phone number in the format `(country code)XXXXXXXXXX`;
  - `verified` – indicates that the phone has been verified – `true` or `false`;
- `email` – an email address in the form of a composite object with attributes:
  - `value` – email address;
  - `verified` – indicates that the address has been verified – `true` or `false`;

### user.credentials block

Optional block.

- `password` is the password for the user account being created (must match the configured password policy).

### actions block

Optional block.

Actions performed after account registration:

- `bindDynClient` - after registering an account, it is necessary to associate with it the previously released free dynamic `client_id` of the mobile application instance.

It is used when registering a user from a mobile application.

Parameters:

- `type` is the name of the action. The value `bindDynClient` must be passed;
- `client_id` is a value containing a dynamic `client_id`.

```
"actions": [
  {
    "type": "bindDynClient",
    "client_id": "dyn~test_app~af...59"
  }
]
```



## Examples

### Registration with a confirmed email and phone number

#### Request

```
PUT /blitz/reg/api/v3/users HTTP/1.1
Authorization: Bearer cNw...Nz
Content-Type: application/json

{
  "user": {
    "attrs": {
      "sub": "BIP-9TZYWXQ",
      "family_name": "Иванов",
      "given_name": "Иван",
      "middle_name": "Иванович",
      "email": {
        "value": "ivan.ivanov@example.com",
        "verified": true
      },
      "phone_number": {
        "value": "79991234567",
        "verified": true
      }
    },
    "credentials": {
      "password": "Qwerty_123"
    }
  }
}
```

#### Response

```
{
  "instanceId": "Yml...Yw",
  "subject": "BIP-9TZYWXQ",
  "context": "M0E...pQ",
  "cookies": [
    {
      "name": "css",
      "value": "cp0...1o"
    }
  ],
  "instructions": []
}
```

## Errors

Listing 11: The password does not comply with the password policy

```
{
  "errors": [
    {
      "errMsg": "Пароль не соответствует парольным политикам: длина менее 8_
↔символов, не содержит цифру, прописную букву, специальный символ.",
      "field": "password"
    }
  ],
  "context": ""
}
```

Listing 12: The uniqueness of the fields is violated

```
{
  "errors": [
    {
      "errMsg": "Пользователь с таким значением уже зарегистрирован. Для_
↔дальнейшей регистрации введите другое значение",
      "field": "phone_number"
    },
    {
      "errMsg": "Пользователь с таким значением уже зарегистрирован. Для_
↔дальнейшей регистрации введите другое значение",
      "field": "email"
    },
    {
      "errMsg": "Пользователь с таким значением уже зарегистрирован. Для_
↔дальнейшей регистрации введите другое значение",
      "field": "sub"
    }
  ],
  "context": ""
}
```

## Registration with an unconfirmed email and phone number

## Request

```
PUT /blitz/reg/api/v3/users HTTP/1.1
Authorization: Bearer cNw...Nz
Content-Type: application/json

{
  "user": {
    "attrs": {
      "sub": "BIP-1TZYWXQ",
      "family_name": "Иванов",
      "given_name": "Иван",
      "middle_name": "Иванович",
      "email": {
        "value": "ivan.ivanov@example.com",
        "verified": false
      },
      "phone_number": {
```

(continues on next page)

(continued from previous page)

```

        "value": "79991234567",
        "verified": false
      },
    },
    "credentials": {
      "password": "Qwerty_123"
    }
  }
}

```

### Response No.1

If registration is caused by the transmission of an unconfirmed phone and/or email, the service will send the user a verification SMS with a confirmation code and/or email with a confirmation code and return the service attributes `instructions` and `context`.

The response is when the user needs to enter verification codes:

```

{
  "context": "NIi...qQ",
  "instructions": [
    {
      "mobile": "+79991234567",
      "exp": 1690444604,
      "attempts": 3,
      "name": "mbl-enter-code"
    },
    {
      "email": "ivan.ivanov@example.com",
      "exp": 1690644970,
      "attempts": 3,
      "name": "eml-enter-code"
    }
  ]
}

```

The registration service can be configured so that the user is registered immediately, and contacts are registered in the account after confirmation, in this case, the registration service will return the parameters of the registered account (`instanceId`, `subject`, `cookies`), as well as instructions for optional confirmation of contacts in the account:

```

{
  "instanceId": "Yml...Yw",
  "subject": "BIP-1TZYWXQ",
  "context": "NIi...qQ",
  "cookies": [
    {
      "name": "css",
      "value": "t8_...84"
    }
  ],
  "instructions": [
    {
      "mobile": "+79991234567",
      "exp": 1690444604,
      "attempts": 3,
      "name": "mbl-enter-code"
    },
  ],
}

```

(continues on next page)

(continued from previous page)

```

    {
      "email": "ivan.ivanov@example.com",
      "exp": 1690644970,
      "attempts": 3,
      "name": "eml-enter-code"
    }
  ]
}

```

### Confirmation codes

When receiving the instructions `eml-enter-code` and/or `mbl-enter-code` in response No. 1, you need to ask the user to enter the confirmation code sent to email and mobile phone. After entering each code, call the service to confirm the contact specified during registration by passing the value from the `context` parameter to the request URL, and the confirmation code entered by the user in the request body:

Listing 13: Email confirmation request

```

POST /blitz/reg/api/v3/users/YNx9...Dw HTTP/1.1
Authorization: Bearer cNw...Nz
Content-Type: application/json

{
  "email_code": "269302"
}

```

Listing 14: The response if the wrong code is entered from the email

```

{
  "instructions": [
    {
      "email": "mail123@example.com",
      "exp": 1655283696,
      "attempts": 2,
      "name": "eml-try-again"},
    {
      "mobile": "79988984169",
      "exp": 1655280756,
      "attempts": 3,
      "name": "mbl-try-again"
    }
  ],
  "context": "kE6r...7g"
}

```

Listing 15: Response if the expiration date has expired or the number of attempts has been exceeded (there will be a general error `eml-expired`)

```

{
  "instructions": [
    {
      "email": "mail123@example.com",
      "name": "eml-expired"
    },
    {
      "mobile": "79988984169",

```

(continues on next page)

(continued from previous page)

```

        "exp":1655280756,
        "attempts":3,"name":"mbl-try-again"
    }
  ],
  "context":"kE6r...7g"
}

```

Listing 16: Request to initiate the re-sending of the code by email (specify any code as the parameter value)

```

POST /blitz/reg/api/v3/users/YNx9...Dw HTTP/1.1
Authorization: Bearer cNw...Nz
Content-Type: application/json

{
  "email_code_resend":"123456"
}

```

If the email has been successfully confirmed, and it remains to confirm the phone, then the instructions about confirming the email will disappear in the service's response, and only the instructions about the phone will remain:

Listing 17: Response if the email is confirmed, but you need to confirm the phone number

```

{
  "instructions": [
    {
      "mobile":"79988984169",
      "exp":1655280756,
      "attempts":3,
      "name":"mbl-try-again"
    }
  ],
  "context":"kE6r...7g"
}

```

Listing 18: Phone number confirmation request

```

POST /blitz/reg/api/v3/users/YNx9...Dw HTTP/1.1
Authorization: Bearer cNw...Nz
Content-Type: application/json

{
  "sms_code":"953568"
}

```

Listing 19: Response if the wrong phone verification code is entered

```

{
  "instructions": [
    {
      "email":"mail123@example.com",
      "exp":1655283696,
      "attempts":2,
      "name":"eml-try-again"},
    {

```

(continues on next page)

(continued from previous page)

```

        "mobile": "799888984169",
        "exp": 1655280756,
        "attempts": 3,
        "name": "mbl-try-again"
    }
  ],
  "context": "kE6r...7g"
}

```

Listing 20: Response if the expiration date has expired

```

{
  "instructions": [
    {
      "mobile": "799888984169",
      "name": "mbl-expired"
    }
  ],
  "context": "kE6r...7g"
}

```

Listing 21: Response if the number of attempts is exceeded

```

{
  "instructions": [
    {
      "mobile": "799888984169",
      "name": "mbl-no-attempts"
    }
  ],
  "context": "kE6r...7g"
}

```

Listing 22: Request to initiate the re-sending of the code via SMS (specify any code as the parameter value)

```

POST /blitz/reg/api/v3/users/YNx9...Dw HTTP/1.1
Authorization: Bearer cNw...Nz
Content-Type: application/json

{
  "sms_code_resend": "123456"
}

```

## Response No.2

If all contacts were confirmed during the registration process, then as a result of calling the service, a user account with the provided attributes and password will be registered in Blitz Identity Provider. The service will return the user ID assigned to the account (`subject`). In addition, a number of service attributes (`instructions`, `cookies` and `context`) will be returned.

```

{
  "instanceId": "Yml...Yw",
  "subject": "BIP-1TZYWXQ",
  "context": "NIi...qQ",
  "cookies": [

```

(continues on next page)

(continued from previous page)

```

    {
      "name": "css",
      "value": "t8_...84"
    }
  ],
  "instructions": []
}

```

## Error

Registration may fail. Then there will be an explanation of the problem in the body of the response. In particular, if the uniqueness of an attribute is violated in Blitz Identity Provider, the message will contain a list of fields for which uniqueness is violated.

```

{
  "errors": [
    {
      "errMsg": "Такой пользователь уже зарегистрирован...",
      "field": "email"
    },
    {
      "errMsg": "Такой пользователь уже зарегистрирован...",
      "field": "phone_number"
    }
  ],
  "context": ""
}

```

## Registration with a confirmed email and phone number with the transfer of a dynamic client\_id

Listing 23: Request

```

PUT /blitz/reg/api/v3/users HTTP/1.1
Authorization: Bearer cNw...Nz
Content-Type: application/json

{
  "user": {
    "attrs": {
      "sub": "BIP-9TZYWXQ",
      "family_name": "Иванов",
      "given_name": "Иван",
      "middle_name": "Иванович",
      "email": {
        "value": "ivan.ivanov@example.com",
        "verified": true
      },
      "phone_number": {
        "value": "79991234567",
        "verified": true
      }
    },
    "credentials": {
      "password": "Qwerty_123"
    }
  },
}

```

(continues on next page)

(continued from previous page)

```

"actions": [
  {
    "type": "bindDynClient",
    "client_id": "dyn~test-app~c84f26f3-10f3-4b85-a6ee-a4ca12c41d26"
  }
]
}

```

## Registration in English

Listing 24: Request

```

curl -v --location --request PUT 'https://demo.identityblitz.com/blitz/reg/api/v3/
↪users' \
--header 'Content-Type: application/json' \
--header 'Accept-Language: en' \
--header 'Authorization: Bearer ...' \
--data-raw '{
  "user": {
    "attrs": {
      "sub": "username",
      "phone_number": {
        "value": "89101234567",
        "verified": false
      }
    },
    "credentials": {
      "password": "Qwerty_123"
    }
  }
}'

```

## Search

Method GET <https://login.company.com/blitz/api/v1/users>

Search for an account.

URL parameters A search query in [Resource Query Language<sup>84</sup>](#) (RQL) format is passed to query. Operations:

- and - simultaneous execution of search conditions;
- or - alternative fulfillment of search conditions (for example, search by different attributes);
- eq - checking the equality condition.

When searching for an attribute with a string value, it is recommended to explicitly specify the value type. For example, `string:02142527602`.

**Attention:** If the search attribute is a string containing special characters such as `&` `|` `()` `=<>` `,` `,` then it is necessary to adhere to the following algorithm for escaping and encoding parameters:

1. To encode all attribute values – to escape the special characters present in the parameters. For example, if you are searching by phone `+7(999)1234567`, then the parameter value should be converted to the value `+7%28999%291234567`.

<sup>84</sup> <https://github.com/kriszyp/rql>



2. Assemble a common string to pass as a query parameter to the query. For example, `phone_number=+7%28999%291234567`.
3. Execute the URL Encode of the parameter value. For example, the parameter value is `phone_number%3D%2B7%2528999%25291234567`.

## Examples

### Simple search query

#### Request

```
GET /blitz/api/v1/users?query=eq(phone_number.string:79991234567) HTTP/1.1
Authorization: Basic YXBwX2lkOmFwcF9zZWNYZXQ=
```

#### Response

```
[
  {
    "instanceId": "Mzg5...nU",
    "attrs": {
      "sub": "854436f6-af58-4a3f-8cb7-c2c441eb4a76",
      "family_name": "Иванов",
      "given_name": "Иван",
      "middle_name": "Иванович",
      "phone_number": "79991234567",
    }
  }
]
```

### Complex search query

#### Listing 25: Request

```
GET /blitz/api/v1/users?query=or(eq(phone_number,string:79991234567),eq(phone_
↵number,string:79991112233)) HTTP/1.1
Authorization: Basic YXBwX2lkOmFwcF9zZWNYZXQ=
```

### Search for a string containing special characters

## Listing 26: Request

```
GET /blitz/api/v1/users?query=phone_number%3D%2B7%2528999%25291234567 HTTP/1.1
Authorization: Basic YXBwX2lkOmFwcF9zZWNYZXQ=
```

### Attributes

#### Getting attributes

Method GET `https://login.company.com/blitz/api/v3/users/{subjectId}`

Getting attributes of any user by his ID.

Required permissions: `blitz_api_user` or `blitz_api_sys_users`.

Returns JSON containing user attributes. The metadata of the account is transmitted in the `meta` block.

---

**Important:** The `instanceId` attribute of metadata is needed to be able to call the following services in the future for [account attribute modification](#) (page 358) and a [password change](#) (page 366).

---

### Example

#### Request

```
GET /blitz/api/v3/users/d2580c98-e584-4aad-a591-97a8cf45cd2a HTTP/1.1
Authorization: Bearer cNw...Nz
```

#### Response

```
{
  "family_name": "Иванов",
  "sub": "d2580c98 e584 4aad a591 97a8cf45cd2a",
  "given_name": "Иван",
  "locked": false,
  "meta": {
    "instanceId": "Mzg...J1",
    "unmodifiable": [
      "sub"
    ]
  }
}
```

## Changing an attribute

Method `POST https://login.company.com/blitz/api/v3/users/{instanceId}`

Changing user attributes by `instanceId`. To find out the value of `instanceId`, you must first use the GET method to call the service for [getting the user attributes](#) (page 357).

Required permissions: `blitz_api_user_chg` or `blitz_api_sys_users_chg`.

Request body The values of the user attributes that are being changed.

Returns JSON containing user attributes.

If the passed attribute values did not pass verification, the error `HTTP 400 Bad Request` will return and the nested JSON including:

- the error type is `input_error` for cases when the request contains an incorrect or invalid value;
- error code (error);
- a text description of the error.

**Note:** Error codes and error texts can be defined specifically for various attributes and determined by the logic of validators implemented for attributes.

## Example

### Request

```
POST /blitz/api/v3/users/Mzg...J1 HTTP/1.1
Authorization: Bearer cNw...Nz
Content-Type: application/json

{
  "family_name": "Петров"
}
```

### Response

```
{
  "family_name": "Петров",
  "given_name": "Иван",
  "locked": false,
  "sub": "5cffd68f-2cb8-4f7a-b0f3-9fa69a1fbbcd",
  "meta": {
    "instanceId": "Mzg...J1",
    "unmodifiable": [
      "sub"
    ]
  }
}
```

## Error

```
{
  "type": "input_error",
  "error": "wrong_values",
  "errors": [
    {
      "type": "input_error",
      "error": "contact_use_violation",
      "desc": "Validation mobile:79988887812 is failed.",
      "pos": "mobile"
    }
  ]
}
```

## Changing the phone number

Method Special case of *attribute modification* (page 358).

Modes:

- changing the phone number immediately to a confirmed one,
- changing the phone number with confirmation..

Request body

- `phone_number` is a mobile phone, in the form of a composite object with attributes:
  - `value` is a phone number in the format (country code)XXXXXXXXXX;
  - `vrf` – indicates that the phone has been confirmed – true.

## Examples

### Changing the number to a confirmed one

#### Request

```
POST /blitz/api/v3/users/Mzg...J1 HTTP/1.1
Authorization: Bearer wzb...Tw
Content-Type: application/json

{
  "phone_number":
    {
      "value": "79991234567",
      "vrf": true
    }
}
```

## Response

```
{
  "given_name": "Иван",
  "family_name": "Иванов",
  "meta": {
    "instanceId": "Mzg5L...2M",
    "unmodifiable": [
      "uid"
    ]
  },
  "email": {
    "value": "aivanov+2@gmail.com",
    "vrf": true
  },
  "sub": "BIP-LIR6BO33XBBDHANE6DZPUTYVME",
  "phone_number": {
    "value": "+7(999)1234567",
    "vrf": true
  }
}
```

## Changing the number with confirmation

### Request

```
POST /blitz/api/v3/users/Mzg...J1 HTTP/1.1
Content-Type: application/json
X-Forwarded-For: 200.200.100.100
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_5) AppleWebKit/537.36_
→ (KHTML, like Gecko) Chrome/83.0.4103.106 Safari/537.36
Authorization: Bearer wzb...Tw

{
  "phone_number": {"value": "+799999999998", "vrf": false}
}
```

### Response No. 1

The interim response contains an indication of the need to confirm a new phone number. The confirmation code is sent to the user at the new number.

```
{
  "given_name": "Иван",
  "family_name": "Иванов",
  "meta": {
    "instanceId": "Mzg5L...2M",
    "unmodifiable": [
      "sub"
    ]
  },
  "email": {
    "value": "aivanov+2@gmail.com",
    "vrf": true
  },
  "sub": "BIP-LIR6BO33XBBDHANE6DZPUTYVME",
  "notes": {
```

(continues on next page)

(continued from previous page)

```

    "actions": {
      "state": "ch_EludIw5fEDouy8wpT_GVOJ7rLxKfZUi-G3blijf34yQ",
      "exp": 300,
      "status": "code_waiting",
      "from": "+7(964)1234567",
      "attr": "mobile",
      "attempts_left": 3,
      "value": "+7(999)9999998",
      "action": "validate_mobile",
      "created": 1598446512
    },
    "phone_number": {
      "value": "+7(964)1234567",
      "vrf": true
    }
  }
}

```

### Confirmation code

You need to get a confirmation code for the new phone number from the user and send it to Blitz Identity Provider in the request. In the URL of this request, use the value of the `actions: state` parameter from response No. 1:

```

POST /blitz/api/v3/users/notes/validate_mobile/ch_El...yQ HTTP/1.1
Content-Type: application/json
X-Forwarded-For: 200.200.100.100
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_5)...
Authorization: Bearer wzb...Tw

{
  "cmd": "code",
  "value": "123456"
}

```

### Response No. 2

Listing 27: Successful phone number change

```

{
  "given_name": "Иван",
  "family_name": "ИВАНОВ",
  "meta": {
    "instanceId": "Mzg5L...2M",
    "unmodifiable": [
      "sub"
    ]
  },
  "email": {
    "value": "aivanov+2@gmail.com",
    "vrf": true
  },
  "sub": "BIP-LIR6B033XBBDHANE6DZPUTYVME",
  "phone_number": {
    "value": "+7(999)9999998",
    "vrf": true
  }
}

```

## Error

Listing 28: Invalid code

```
{
  "state": "ch_EludIw5fEDouy8wpT_GVOJ7rLxKfZUi-G3blijf34yQ",
  "exp": 2592000,
  "from": "+7 (964) 1234567",
  "attr": "phone_number",
  "msg": "wrong_code",
  "attempts_left": 2,
  "created": 1649695409,
  "value": "+7 (999) 9999998",
  "action": "validate_mobile"
}
```

Listing 29: Exceeded the number of attempts to enter the correct code

```
{
  "state": "ch_EludIw5fEDouy8wpT_GVOJ7rLxKfZUi-G3blijf34yQ",
  "id": "ch_EludIw5fEDouy8wpT_GVOJ7rLxKfZUi-G3blijf34yQ",
  "attr": "phone_number",
  "cause": "no_attempts_left",
  "from": "+7 (964) 1234567",
  "value": "+7 (999) 9999998",
  "action": "validate_mobile"
}
```

Listing 30: The code is expired

```
{
  "state": "ch_EludIw5fEDouy8wpT_GVOJ7rLxKfZUi-G3blijf34yQ",
  "id": "ch_EludIw5fEDouy8wpT_GVOJ7rLxKfZUi-G3blijf34yQ",
  "attr": "phone_number",
  "cause": "code_expired",
  "from": "+7 (964) 1234567",
  "value": "+7 (999) 9999998",
  "action": "validate_mobile"
}
```

## Changing the email address

Method Special case of *attribute modification* (page 358).

Modes:

- changing the email immediately to a confirmed one,
- changing email with confirmation.

Request body

- email – email address:
  - value – email address;
  - vrf – indicates that the address has been confirmed – true;

## Examples

### Changing the address to a confirmed one

#### Request

```
POST /blitz/api/v3/users/Mzg...J1 HTTP/1.1
Authorization: Bearer wzb...Tw
Content-Type: application/json

{
  "email":
    {
      "value": "mail@example.com",
      "vrf": true
    }
}
```

#### Response

```
{
  "given_name": "Иван",
  "family_name": "Иванов",
  "meta": {
    "instanceId": "Mzg5LW...2M",
    "unmodifiable": [
      "sub"
    ]
  },
  "mail": {
    "value": "mail@example.com",
    "vrf": true
  },
  "sub": "BIP-LIR6BO33XBBDHANE6DZPUTYVME",
  "phone_number": {
    "value": "+7 (999) 1234567",
    "vrf": true
  }
}
```

### Address change with confirmation

#### Request

```
POST /blitz/api/v3/users/Mzg...J1 HTTP/1.1
Content-Type: application/json
X-Forwarded-For: 200.200.100.100
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_5) AppleWebKit/537.36_
↔ (KHTML, like Gecko) Chrome/83.0.4103.106 Safari/537.36
Authorization: Bearer wzb...Tw

{
  "email": {"value": "mail@example.com", "vrf": false}
}
```



### Response No. 1

The interim response contains an indication of the need to confirm the new email address. The confirmation code is sent to the user at the new address.

```
{
  "given_name": "Иван",
  "family_name": "Иванов",
  "meta": {
    "instanceId": "Mzg5L...2M",
    "unmodifiable": [
      "sub"
    ]
  },
  "email": {
    "value": "aivanov+2@gmail.com",
    "vrf": true
  },
  "sub": "BIP-LIR6BO33XBBDHANE6DZPUTYVME",
  "notes": {
    "actions": {
      "state": "ch_EludIw5fEDouy8wpT_GVOJ7rLxKfZUi-G3blijf34yQ",
      "exp": 86400,
      "status": "code_waiting",
      "from": "aivanov+2@gmail.com",
      "attr": "mail",
      "attempts_left": 3,
      "value": "mail@example.com",
      "action": "validate_mail",
      "created": 1598446512
    }
  },
  "phone_number": {
    "value": "+7(964)1234567",
    "vrf": true
  }
}
```

### Confirmation code

You need to get a confirmation code for the new email address from the user and send it to Blitz Identity Provider in the request. In the URL of this request, use the value of the `actions: state` parameter from response No. 1:

```
POST /blitz/api/v3/users/notes/validate_email/ch_El...yQ HTTP/1.1
Content-Type: application/json
X-Forwarded-For: 200.200.100.100
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_5)...
Authorization: Bearer wzb...Tw

{
  "cmd": "code",
  "value": "123456"
}
```

## Response No. 2

Listing 31: Successful email address change

```
{
  "given_name": "Иван",
  "family_name": "Иванов",
  "meta": {
    "instanceId": "Mzg5L...2M",
    "unmodifiable": [
      "sub"
    ]
  },
  "email": {
    "value": "mail@example.com",
    "vrf": true
  },
  "sub": "BIP-LIR6B033XBBDHANE6DZPUTYVME",
  "phone_number": {
    "value": "+7(999)9999998",
    "vrf": true
  }
}
```

## Error

Listing 32: Invalid code

```
{
  "state": "ch_EludIw5fEDouy8wpT_GVOJ7rLxKfZUi-G3blijf34yQ",
  "exp": 2592000,
  "from": "aivanov+2@gmail.com",
  "attr": "email",
  "msg": "wrong_code",
  "attempts_left": 2,
  "created": 1649695409,
  "value": "mail@example.com",
  "action": "validate_email"
}
```

Listing 33: Exceeded the number of attempts to enter the correct code

```
{
  "state": "ch_EludIw5fEDouy8wpT_GVOJ7rLxKfZUi-G3blijf34yQ",
  "id": "ch_EludIw5fEDouy8wpT_GVOJ7rLxKfZUi-G3blijf34yQ",
  "attr": "email",
  "cause": "no_attempts_left",
  "from": "aivanov+2@gmail.com",
  "value": "mail@example.com",
  "action": "validate_email"
}
```

Listing 34: The code is expired

```
{
  "state": "ch_EludIw5fEDouy8wpT_GVOJ7rLxKfZUi-G3blijf34yQ",
  "id": "ch_EludIw5fEDouy8wpT_GVOJ7rLxKfZUi-G3blijf34yQ",
```

(continues on next page)

(continued from previous page)

```

"attr": "email",
"cause": "code_expired",
"from": "aivanov+2@gmail.com",
"value": "mail@example.com",
"action": "validate_email"
}

```

## Passwords

### Changing the password

Method `POST https://login.company.com/blitz/api/v3/users/{instanceId}/pswd`

Password change. To find out the value of the `instanceId` for the user, you must first call the service for [getting the user attributes](#) (page 357) with the GET method.

Required permissions: `blitz_api_usec_chg` or `blitz_api_sys_usec_chg`.

#### Headers

- When changing the password in user mode, you need to transmit headers with the user's IP address and `User-Agent`.
- In the scenario of the user changing the password independently in the User Profile, it is possible to reset the user's sessions. In this case, it may be undesirable for the user to log out of the current device/browser. In order to specify Blitz Identity Provider that a certain device must be saved based on the results of a successful password change (do not log out from it), you need to transfer the `IB-CI-UA-ID` header with the identifier of the current user device from the application to the password change service call.

---

**Tip:** The ID of the user's current device can be obtained from the [identification token](#) (page 309).

---

- To send an e-mail in English, specify the `Accept-Language: en` header (available only in v3).

#### Request body

- `current` – the user's current password (only when changing the password in user mode, it must be transmitted).
- `password` is the user's new password (optional parameter). If the parameter is omitted, Blitz Identity Provider will generate a new password on its own.
- `resetSessions` – if the parameter is not specified or is set to `true`, then when changing the password, all user sessions will be canceled and the stored devices will be deleted. If you only need to change the password without resetting sessions, then you must explicitly specify the parameter in the value `false`.
- `sendPswdToAttr` is the name of the attribute with the phone number to send the password to the user (optional parameter). If the parameter is set, an SMS with a password will be sent to the user's phone from the specified attribute.

#### Returns

- In case of a successful call to Blitz Identity Provider - HTTP 204 No Content.
- If the password change failed, an error message is displayed:
  - HTTP 401 Unauthorized in case of an access control error, the access token is incorrect or the user's current password is incorrect.
  - HTTP 400 Bad Request - the new password does not meet the requirements of the password policy.

## Examples

### Request

Listing 35: Custom password change mode

```
POST /blitz/api/v3/users/Mzg...J1/pswd HTTP/1.1
Content-Type: application/json
X-Forwarded-For: 200.200.100.100
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_5)...
Authorization: Bearer wzb...Tw
IB-CI-UA-ID: {SHA256}rVWFmwgRKWeW_flH4CA4yuW7OhKZ32Da94m0kzwWsVs

{
  "current": "QWErty123",
  "password": "P@$$w0rd",
  "resetSessions": false
}
```

Listing 36: Password change mode by the system

```
POST /blitz/api/v3/users/Mzg...J1/pswd HTTP/1.1
Content-Type: application/json
Authorization: Bearer qwa...Ez

{
  "password": "P@$$w0rd",
  "resetSessions": true
}
```

Listing 37: Sending a new password via SMS with automatic password generation

```
POST /blitz/api/v3/users/Mzg...J1/pswd HTTP/1.1
Content-Type: application/json
Authorization: Bearer qwa...Ez

{
  "sendPswdToAttr": "phone_number"
}
```

Listing 38: Password change request in English

```
curl -v --location --request POST 'https://demo.identityblitz.com/blitz/api/v3/
→users/YnVpbHQtaW46a2dhdnJpbG92QG1kYmxpdHoucU6MTcxMDU5ODgyODY3MjU0ODg2NA/pswd' \
--header 'Content-Type: application/json' \
--header 'Accept-Language: en' \
--header 'Authorization: Bearer ...' \
--data-raw '{"password": "nN2L98Nu1234"}'
```

## Errors

Listing 39: Incorrect current password

```
{
  "type": "security_error",
  "error": "invalid_credential",
  "desc": "Wrong subject identifier or current password"
}
```

Listing 40: Incorrect access token

```
{
  "type": "security_error",
  "error": "bad_access_token",
  "desc": "BEARER_AUTH: CRID does not match"
}
```

Listing 41: The new password does not comply with the password policy:  
too short

```
{
  "type": "input_error",
  "error": "wrong_values",
  "errors": [
    {
      "type": "input_error",
      "error": "password_policy_violated",
      "desc": "Password's length must be greater than 6",
      "pos": "password",
      "params": {
        "rule": "to_short",
        "low": 6
      }
    }
  ]
}
```

Listing 42: The new password does not comply with the password policy  
set in the LDAP directory

```
{
  "type": "input_error",
  "error": "password_policy_violated",
  "desc": "Failed to update password\n",
  "pos": "password",
  "params": {
```

(continues on next page)

(continued from previous page)

```

    "rule": "id_store"
  }
}

```

Listing 43: The new password does not comply with the password policy: does not contain the required character groups

```

{
  "type": "input_error",
  "error": "wrong_values",
  "errors": [
    {
      "type": "input_error",
      "error": "password_policy_violated",
      "desc": "Password doesn't match enough symbols groups",
      "pos": "password",
      "params": {
        "rule": "not_enough_groups",
        "no_matched_groups": [
          {
            "desc": "password.policy.desc.digits",
            "min_number_symbols": 1
          },
          {
            "desc": "password.policy.desc.capital",
            "min_number_symbols": 1
          },
          {
            "desc": "password.policy.desc.special",
            "min_number_symbols": 1
          }
        ]
      }
    }
  ]
}

```

Listing 44: The new password does not comply with the password policy: the password was previously used

```

{
  "type": "input_error",
  "error": "wrong_values",
  "errors": [
    {
      "type": "input_error",
      "error": "password_policy_violated",
      "desc": "Password found in previous used ones",
      "pos": "password",
      "params": {
        "rule": "in_password_history"
      }
    }
  ]
}

```

Listing 45: The new password does not comply with the password policy:  
the new password matches the current one

```
{
  "type": "input_error",
  "error": "wrong_values",
  "errors": [
    {
      "type": "input_error",
      "error": "password_policy_violated",
      "desc": "A new password can't be the same as the current",
      "pos": "password",
      "params": {
        "rule": "eq_current"
      }
    }
  ]
}
```

Listing 46: The new password does not comply with the password policy:  
in the new password, the insufficient number of characters differs from  
the previous one

```
{
  "type": "input_error",
  "error": "wrong_values",
  "errors": [
    {
      "type": "input_error",
      "error": "password_policy_violated",
      "desc": "There are not enough new characters in a new password",
      "pos": "password",
      "params": {
        "rule": "not_enough_new_chars",
        "minNew": 5
      }
    }
  ]
}
```

Listing 47: The new password does not comply with the password policy:  
the password includes an entry from the dictionary of prohibited  
passwords

```
{
  "type": "input_error",
  "error": "wrong_values",
  "errors": [
    {
      "type": "input_error",
      "error": "password_policy_violated",
      "desc": "Password contains a word from the stop dictionary",
      "pos": "password",
      "params": {
        "rule": "in_stop_dic",
        "stop_word": "qwerty"
      }
    }
  ]
}
```

(continues on next page)

(continued from previous page)

}

Listing 48: The new password does not comply with the password policy:  
the password matches the dictionary password

```
{
  "type": "input_error",
  "error": "wrong_values",
  "errors": [
    {
      "type": "input_error",
      "error": "password_policy_violated",
      "desc": "Password found in a password dictionary",
      "pos": "password",
      "params": {
        "rule": "in_password_dic"
      }
    }
  ]
}
```

Listing 49: The new password does not comply with the password policy:  
the password was changed earlier than the allowed period

```
{
  "type": "input_error",
  "error": "wrong_values",
  "errors": [
    {
      "type": "input_error",
      "error": "password_policy_violated",
      "desc": "Password is too young",
      "pos": "password",
      "params": {
        "rule": "too_young",
        "minAgeInSec": 86400
      }
    }
  ]
}
```

Listing 50: The passed attribute for sending the password does not exist

```
{
  "type": "input_error",
  "error": "wrong_values",
  "errors": [
    {
      "type": "input_error",
      "error": "wrong_value",
      "desc": "Wrong mobile attribute 'phone_number_wrong'",
      "pos": "sendPswdToAttr"
    }
  ]
}
```



Listing 51: The user does not have a phone attribute set to send the password to the phone

```
{
  "type": "input_error",
  "error": "wrong_values",
  "errors": [
    {
      "type": "input_error",
      "error": "wrong_value",
      "desc": "User not contains mobile attribute 'phone_number'",
      "pos": "sendPswdToAttr"
    }
  ]
}
```

### Changing the password of subordinate account

**Method** POST `https://login.company.com/blitz/api/v2/users/{subjectId}/password`

Changing the password of the managed user account using the master user account. `subjectId` is the identifier (sub) of the managed account.

**Headers** A header with a permission access token named `blitz_change_password` received by the lead account should be added to the request. The lead user can trigger a change of the subordinate account password only if the previously lead user *was given* (page 413) the right to change the password `change_password`.

**Request body** The `value` attribute with the value of the new password, which must meet the requirements of the configured password policy.

**Returns**

- If the password is changed successfully, the status is HTTP 200 (OK).
- If there is an error, a description of the error received.

### Example

#### Request

```
POST /blitz/api/v2/users/c574a512-3704-4576-bc3a-3fe28b636e85/password HTTP/1.1
Authorization: Bearer cNwIX...Tg
Content-Type: application/json

{"value": "QWErty1234"}
```

## Error

```
{
  "errors": [
    {
      "code": "access_denied",
      "desc": "Not enough rights: change_password",
      "params": {}
    }
  ]
}
```

## Authentication modes

### Checking the status

Method GET `https://login.company.com/blitz/api/v3/users/{subjectId}/auth`

Checking the status of the following authentication modes of the *SubjectID* account:

- two-factor authentication enabled;
- the presence of an established indication of the need to change the password;
- the presence of a temporary ban on login using a certain login method.

Required permissions: `blitz_api_usec` or `blitz_api_sys_usec`.

Returns

- `requiredFactor` indicates that two-factor authentication is enabled. It can take the following values:
  - missing, 0 or 1 - disabled,
  - 2 - enabled (2nd authentication factor is required);
- `needPasswordChange` indicates the need to change the password when logging in;
- `methodsLocked` is a list of blocked authentication methods. The user cannot use these login methods, but can use the rest.

## Example

### Request

```
GET /blitz/api/v3/users/d2580c98-e584-4aad-a591-97a8cf45cd2a/auth HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqLOFWDuwzMDc0Nz
Cache-Control: no-cache
```

### Response

```
{
  "requiredFactor": 2,
  "needPasswordChange": true,
  "methodsLocked": ["password"]
}
```

## Changing authentication modes

POST `https://login.company.com/blitz/api/v3/users/{subjectId}/auth`

Changes to user authentication modes.

Required permissions: `blitz_api_usecg` or ``blitz_api_sys_usecg`.

Headers In user mode, headers with the user's IP address and `User-Agent` must be passed.

Request body It may contain parameters:

- `requiredFactor` indicates that two-factor authentication is enabled. Values:
  - `null` is disabled,
  - `2` is enabled (2nd authentication factor is required);
- `needPasswordChange` indicates the need to change the password when logging in – only passing the value `true` is allowed;
- `methodsLocked` is a list of blocked authentication methods. The user cannot use these login methods, but can use the rest. Currently, Blitz Identity Provider only supports blocking the use of password login (`password`).

## Example

### Request

```
POST /blitz/api/v3/users/d2580c98-e584-4aad-a591-97a8cf45cd2a/auth HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqLOFWDuwzMDc0Nz
X-Forwarded-For: 200.200.100.100
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_5)...
Content-Type: application/json

{
  "requiredFactor": 2,
  "needPasswordChange": true,
  "methodsLocked": ["password"]
}
```

### Response

```
{
  "requiredFactor": 2,
  "needPasswordChange": true,
  "methodsLocked": ["password"]
}
```

## Error

Listing 52: HTTP 400 Bad Request: The user has not configured any method for the second authentication factor

```
{
  "type": "input_error",
  "error": "wrong_values",
  "errors": [
    {
      "type": "input_error",
      "error": "has_not_sf_methods",
      "desc": "User 'd2580c98-e584-4aad-a591-97a8cf45cd2a' has not any_
↵second factor method",
      "pos": "requiredFactor"
    }
  ]
}
```

## User properties

### Obtaining properties

Method GET <https://login.company.com/blitz/api/v3/users/{subjectId}/props>

Obtaining properties of a user by user's ID.

Required permissions: `blitz_api_user` or `blitz_api_sys_users`.

Returns HTTP 200 and JSON containing the user's properties.

### Example

#### Request

```
GET /blitz/api/v3/users/854436f6-af58-4a3f-8cb7-c2c441eb4a76/props HTTP/1.1
Content-Type: application/json
Authorization: Bearer cNw...Nz
```

#### Response

```
{
  "pipes.info.fed.readOn":1706530413,
  "fcOn":1707814866,
  "pipes.info.adv-totp.readOn":1696236815,
  "pipes.addKey.mobile.Android.disagreedOn":1701099042,
  "pipes.act.mobile.skippedOn":1695649488,
  "wak.failedOn":1689864670,
  "pipes.act.mobile.outdatedOn":1695649486,
  "last2fa":"x509",
  "pipes.addKey.pc.Windows.disagreedOn":1706100800,
  "pipes.act.mail.skippedOn":1689764346
}
```

## Adding, modifying, and deleting properties

Method `POST https://login.company.com/blitz/api/v3/users/{subjectId}/props`

Adding, modifying, and deleting user properties by user's ID.

Required permissions: `blitz_api_user` or `blitz_api_sys_users`.

Request body JSON with a list of properties to add and delete. To change a value, you need to send the new property value in the `add` section. To delete a property, specify the property to be deleted.

Returns HTTP 200 and JSON containing the actual properties.

### Example

#### Request

Listing 53: Deleting the `last2fa` property and adding `testBool`

```
POST /blitz/api/v3/users/854436f6-af58-4a3f-8cb7-c2c441eb4a76/props HTTP/1.1
Content-Type: application/json
Authorization: Bearer cNw...Nz

{
  "remove" : ["last2fa"],
  "add" : {
    "testBool" : true
  }
}
```

Listing 54: Changing the `testBool` property

```
POST /blitz/api/v3/users/854436f6-af58-4a3f-8cb7-c2c441eb4a76/props HTTP/1.1
Content-Type: application/json
Authorization: Bearer cNw...Nz

{
  "add" : {
    "testBool" : false
  }
}
```

Listing 55: Deleting the testBool property

```

POST /blitz/api/v3/users/854436f6-af58-4a3f-8cb7-c2c441eb4a76/props HTTP/1.1
Content-Type: application/json
Authorization: Bearer cNw...Nz

{
  "remove" : ["testBool"]
}

```

**Response**

Listing 56: Deleting the last2fa property and adding testBool

```

{
  "pipes.act.mobile.skippedOn":1695649488,
  "pipes.act.mobile.outdatedOn":1695649486,
  "testBool":true,
  "pipes.addKey.mobile.Android.disagreedOn":1701099042,
  "pipes.info.adv-totp.readOn":1696236815,
  "wak.failedOn":1689864670,
  "pipes.info.fed.readOn":1706530413,
  "pipes.act.mail.skippedOn":1689764346,
  "fcOn":1707814866,
  "pipes.addKey.pc.Windows.disagreedOn":1706100800
}

```

Listing 57: Changing the testBool property

```

{
  "pipes.act.mobile.skippedOn":1695649488,
  "pipes.act.mobile.outdatedOn":1695649486,
  "testBool":false,
  "pipes.addKey.mobile.Android.disagreedOn":1701099042,
  "pipes.info.adv-totp.readOn":1696236815,
  "wak.failedOn":1689864670,
  "pipes.info.fed.readOn":1706530413,
  "pipes.act.mail.skippedOn":1689764346,
  "fcOn":1707814866,
  "pipes.addKey.pc.Windows.disagreedOn":1706100800
}

```

Listing 58: Deleting the testBool property

```
{
  "pipes.act.mobile.skippedOn":1695649488,
  "pipes.act.mobile.outdatedOn":1695649486,
  "pipes.addKey.mobile.Android.disagreedOn":1701099042,
  "pipes.info.adv-totp.readOn":1696236815,
  "wak.failedOn":1689864670,
  "pipes.info.fed.readOn":1706530413,
  "pipes.act.mail.skippedOn":1689764346,
  "fcOn":1707814866,
  "pipes.addKey.pc.Windows.disagreedOn":1706100800
}
```

## TOTP

**Tip:** See [RFC 6238 TOTP: Time-Based One-Time Password Algorithm](#)<sup>85</sup>.

### Checking for TOTP availability

Method GET <https://login.company.com/blitz/api/v3/users/{subjectId}/totps>

Checking whether the user has a configured TOTP confirmation code generator.

Required permissions: `blitz_api_usec` or `blitz_api_sys_usec`.

Returns If TOTP is configured, its settings will be received in response.

### Example

#### Request

```
GET /blitz/api/v3/users/d2580c98-e584-4aad-a591-97a8cf45cd2a/totps HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqLOFWDuwzMDc0Nz
Cache-Control: no-cache
```

#### Response

```
[
  {
    "id": "SW_TOTP_1_d2580c98-e584-4aad-a591-97a8cf45cd2a",
    "len": 6,
    "name": "Google Authenticator"
  }
]
```

<sup>85</sup> <https://tools.ietf.org/html/rfc6238>

## TOTP linking

Linking to the user account of the TOTP generator is carried out in two stages.

### Stage No.1

**Method** GET `https://login.company.com/blitz/api/v3/users/{subjectId}/totps/attach/qr`

Request for a QR code and a linking string in Blitz Identity Provider.

Required permissions: `blitz_api_usec_chg`or ``blitz_api_sys_usec_chg`.

Headers In user mode, headers with the user's IP address and `User-Agent` must be passed.

Returns Attributes:

- `base64QRCode` is the QR code of the generator linking that needs to be displayed to the user;
- `base32Secret` is a secret generator linking string that needs to be displayed to the user if it is inconvenient for him to photograph the QR code and he prefers to enter the linking code into the generator manually.

### Example

#### Request

```
GET /blitz/api/v3/users/d25..2a/totps/attach/qr HTTP/1.1
Authorization: Bearer cN..z
Cache-Control: no-cache
```

#### Response

```
{
  "base64QRCode": "iVB...g==",
  "base32Secret": "W247OHVTPPTIAOXMGKK6Z7BZ3DEYWO74"
}
```

### Stage No.2

**Method** POST `https://login.company.com/blitz/api/v3/users/{subjectId}/totps/attach/qr`

Confirmation of linking registration.

Required permissions: `blitz_api_usec_chg`or ``blitz_api_sys_usec_chg`.

Request body

- `base32Secret` is the secret initialization string of the TOTP generator;
- `otpCode` is the confirmation code generated by the generator using the TOTP algorithm from the secret string and the current time slot;
- `name` is the display name of the TOTP generator (optional).

Returns

- If successful - HTTP 204 No Content.



- In case of an error, the service - HTTP 400 Bad Request.

### Example

#### Request

```
POST /blitz/api/v3/users/d2580c98..cd2a/totps/SW_TOTP_1_d2580c98..cd2a HTTP/1.1
Content-Type: application/json
X-Forwarded-For: 200.200.100.100
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_5)

{
  "base32Secret": "W247OHVTPPTIAOXMGKK6Z7BZ3DEYWO74",
  "name": "Google Authenticator",
  "otpCode": "123456"
}
```

#### Response

```
{
  "base64QRCode": "iVB...g==",
  "base32Secret": "W247OHVTPPTIAOXMGKK6Z7BZ3DEYWO74"
}
```

#### Error

Listing 59: The wrong code was passed

```
{
  "type": "process_error",
  "error": "wrong_otp_code"
}
```

### Deleting the linking

Method DELETE `https://login.company.com/blitz/api/v3/users/{subjectId}/totps/{id}`

Deleting the linking of the TOTP generator to the user account.

Required permissions: `blitz_api_usec_chg`or ``blitz_api_sys_usec_chg`.

URL parameters The `id` is specified as *received* (page 378) linking ID.

Headers In user mode, headers with the user's IP address and `User-Agent` must be passed.

Returns If successful, the service will return HTTP 204 No Content.

## Example

Listing 60: Request

```
DELETE /blitz/api/v3/users/d..2a/totps/SW_TOTP_1_d..2a HTTP/1.1
Authorization: Bearer cN..z
X-Forwarded-For: 200.200.100.100
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_5)...
```

## Account status

### Checking account status

Method GET https://login.company.com/blitz/api/v3/users/{subjectId}/state

Checking account status:

- presence of blocking due to inactivity;
- presence of a ban on blocking due to inactivity.

Required permissions: blitz\_api\_usec or blitz\_api\_sys\_usec.

## Examples

### Request

```
GET /blitz/api/v3/users/d2580c98-e584-4aad-a591-97a8cf45cd2a/state HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqLOFWDuwzMDc0Nz
Cache-Control: no-cache
```

## Responses

Listing 61: The account status has not been initialized yet (the account has just been created or has not been used before logging in since the function appeared)

```
{
  "name": "initial"
}
```

Listing 62: The account is active

```
{
  "name": "active",
  "checkedOn": 1688106755
}
```

**Note:** The checkedOn parameter stores the timestamp of the last status check.

Listing 63: The account has been blocked due to prolonged inactivity

```
{
  "name": "inactivityLock",
  "on": 1688106646
}
```

**Note:** The `on` parameter stores the blocking time.

Listing 64: The account is in the list of exclusions and cannot be blocked due to inactivity before the date of the `till` parameter

```
{
  "name": "untouchable",
  "till": 1689106755
}
```

**Note:** If the `till` parameter is missing, then the account cannot be blocked at all due to inactivity.

### Changing the account status

Method `POST https://login.company.com/blitz/api/v3/users/{subjectId}/state`

Changing the status of the user account.

Required permissions: `blitz_api_sys_usec_chg`.

Request body Possible parameters:

- `name` is the assigned state. You can only assign the `untouchable` state;
- `till` is an optional parameter in which you can specify the time until which the account is assigned the `untouchable` status. To cancel the `untouchable` status, you can assign the current time to the `till` parameter.

Returns In case of a successful call, HTTP 204 No Content.

### Example

Listing 65: Request

```
POST /blitz/api/v3/users/d2580c98-e584-4aad-a591-97a8cf45cd2a/state HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqLOFDuwzMDc0Nz
Content-Type: application/json

{
  "name": "untouchable",
  "till": 1689106755
}
```

## External providers

### List of external providers

Method `GET /api/v3/users/{subjectId}/fa`

Getting a list of account links of external identity providers to a user account.

Required permissions: `blitz_api_ufa` or `blitz_api_sys_ufa`.

Returns Binding type and name (`fpType` and `fpName`) and the binding identifier (`sid`).

### Example

#### Request

```
GET /blitz/api/v3/users/d2580c98-e584-4aad-a591-97a8cf45cd2a/fa HTTP/1.1
Authorization: Bearer m9tuVBNUizkuwFnq95IXQm1XTplXLUFD105TUmgij4
Cache-Control: no-cache
```

#### Response

```
[
  {
    "sid": "1000347601",
    "fpType": "esia",
    "fpName": "esia_1"
  },
  {
    "sid": "1234",
    "fpType": "tcs",
    "fpName": "tcs_1"
  }
]
```

### Linking a provider by ID

Method `POST /api/v3/users/{subjectId}/fa/{fpType}/{fpName}/{sid}`

Linking the account of an external identity provider to a user account, if logging in through an external identity provider was previously performed by other means and the identifier (`sid`) of the account in the external identity provider is known.

Required permissions: `blitz_api_ufa_chg` or `blitz_api_sys_ufa_chg`.

URL parameters The user's `guid` (`subjectId`), the type of external provider (`fpType`), the name of the external provider (`fpName`) and the account ID in the external provider (`sid`).

Request body JSON:

- `federatedAccountName`: name of the external account to be bound (optional). If the parameter is not passed, the previous name is used.

Returns If the call is successful, 204 No Content.

## Example

Listing 66: Request

```
POST /blitz/api/v3/users/d2580c98-e584-4aad-a591-97a8cf45cd2a/fa/tcs/tcs_1/1234...
↔HTTP/1.1
Authorization: Bearer m9tuVBNUnizkuwFnq95IXQm1XTplXLUFD105TUmGij4

{
  "federatedAccountName": "Elle Woods"
}
```

## Linking a provider

Linking to an external provider account with an unknown account ID in the external provider is carried out in two stages:

- Request for linking instructions.
- Linking by the user in the browser.

Method `POST /api/v2/users/current/fa/bind`

Request for linking instructions.

Request body

- `fp` is the identifier of the provider whose profile should be linked to;
- `callback` is the address to which the user should be returned after successfully linking the social network account;
- `isPopup` – whether the identity provider’s page needs to be opened in the popup window (optional).

Returns The `redirectTo` parameter with a link to which the user must be directed in the browser to complete the second stage and create a linking of the user account to an external identity provider.

## Example

### Request

```
POST /blitz/api/v2/users/current/fa/bind HTTP/1.1
Authorization: Basic ZG5ldm5pay10ZXN0Lm1vcy5ydTphUU56S0JuY2VBQVQwelg
Content-Type: application/json

{
  "fp": "vk:vk_1",
  "callback": "https://app.company.com/callback"
}
```

## Response

```
200 OK
{
  "redirectTo": "https://login.company.com/blitz/api/v2/users/current/fa/bind/auth/
  ↪fc111c86-5193-42a2-862a-d819a4f45a86"
}
```

## Deleting a provider linking

Method DELETE /api/v2/users/{subjectId}/fa/{fpType}/{fpName}/{sid}

Deleting the linking of the external provider to the user.

URL parameters `guid of the user (subjectId)`, `type of external provider (fpType)`, `name of the external provider (fpName)` and *the account ID in the external provider* (page 383) (`sid`).

## Example

Listing 67: Header

```
DELETE /blitz/api/v3/users/d2580c98-e584-4aad-a591-97a8cf45cd2a/fa/tcs/tcs_1/1234_
↪HTTP/1.1
Authorization: Bearer m9tuVBUnizkuwFnq95IXQm1XTplXLUFD105TUmGij4
```

## Obtaining a user access token

Method GET /api/v3/users/\${subjectId}/fedToken/\${fedPointType}/\${fedPointName}

Obtaining a valid user access token in an external identity provider with type `${fedPointType}` and name `${fedPointName}`. An access token is considered valid if its lifetime is greater than the minimum allowed lifetime (30 seconds by default). If an access token is invalid, but it was saved along with an update token, an attempt is made to update the access token. If the attempt is successful, this method produces a new access token.

---

**Important:** Obtaining a token is only possible for those providers that have the `Remember tokens` setting *enabled* (page 109).

---

Required permissions: `fed_tkn_any` or `fed_tkn_${fedPointType}_${fedPointName}`.

**Note:** In order for an application to request an access token, these permissions must be *specified* (page 171) for it as well.

## Returns

- HTTP 404: access token not found.
- HTTP 200 and JSON that contains user access token data in the case of success. For each token, the key `sid`, the token value `token` and the validity period `expiresOn` in Unix-time format are transmitted.
- HTTP 401: no permission or wrong provider.

## Example

### Request

```
GET /blitz/api/v3/users/d2580c98-e584-4aad-a591-97a8cf45cd2a/fedToken/tcs/tcs_1
↪HTTP/1.1
Authorization: Bearer m9tuVBNU nizkuwFnq95IXQm1XTplXLUFD105TUmGij4
Content-Type: application/json
```

### Response

Listing 68: Success

```
{
  "da0c69c5-aef8-41e4-a37f-89c6d30abdfa": {
    "expiresOn": 1711125311,
    "token": "t.eFgoMik6regKsLjxfds1V0PlNEv_smx-W_x"
  },
  "00000000-1111-41e4-a37f-89c6d30abdfa": {
    "expiresOn": 1711125344,
    "token": "t.dddddddddLjxfds1V0PlNEv_smx-W_x"
  }
}
```

Listing 69: No required permission

```
{
  "type": "security_error",
  "error": "bad_access_token",
  "desc": "No enough scopes or wrong subject Id"
}
```

## Audit events

Method GET <https://login.company.com/blitz/api/v3/users/{subjectId}/audit>

Retrieving a list of security events registered to the user's account.

Required permissions: `blitz_api_uaud` or `blitz_api_sys_uaud`.

URL parameters

- `rql` is a request to filter the output information in the format [Resource Query Language<sup>86</sup>](#) (RQL). Filtering by the attribute `ts` (time of the event) is supported.

Operations:

- `and` - simultaneous execution of search conditions;
- `le` - checking the condition "less than or equal to";
- `ge` - checking the condition "greater than or equal to";
- `limit` - a limit on the number of records to be returned.

- `ua` - the required type of output of information about the `UserAgent` (attribute `ua`). Options:
  - `none` - not to return the `UserAgent`;

<sup>86</sup> <https://github.com/kriszyp/rql>

- `parsed` – return the `UserAgent` in disassembled form (separate browser and operating system with their versions);

If the `ua` parameter is omitted, then `UserAgent` (the `ua` attribute) will be returned simply as a string.

Returns JSON containing a list of account audit events for the specified time period.

## Examples

### Without parsing information about UserAgent

#### Request

```
GET /blitz/api/v3/users/d2580c98-e584-4aad-a591-97a8cf45cd2a/audit?rq1=and(ge(ts,
↪1637230238),le(ts,1637250238),limit(2)) HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqLOFWDuwzMDc0Nz
Cache-Control: no-cache
```

#### Response

```
[
  {
    "sbj": "af583e70-fe39-407d-a87e-06cd0ec1830c",
    "ua": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) ...",
    "ts": 1637250238015,
    "cAthM": "Basic",
    "ipCt": "Москва",
    "ipRad": 20,
    "cId": "test_app",
    "ip": 1406987879,
    "obj": "af583e70-fe39-407d-a87e-06cd0ec1830c",
    "ipSt": "Москва",
    "lpId": "test_app",
    "pid": "ddeebaba-2dc3-41bb-b539-7f0e472414a3",
    "ipLat": 55.7483,
    "prms": {
      "used_login": "test@yandex.ru",
      "auth_methods": "password",
      "authnDone": "true",
      "id_store": "389-ds"
    },
    "type": "login",
    "ipCtr": "Россия",
    "proc": "profile",
    "ipLng": 37.6171,
    "sid": "54914ac3-0d39-40d3-9617-92e0e7fe07ab"
  }
]
```



## With parsing information about UserAgent

### Request

```
GET /blitz/api/v3/users/d2580c98-e584-4aad-a591-97a8cf45cd2a/audit?rq1=and (ge (ts,
↪1637230238), le (ts, 1637250238), limit (2)) &ua=parsed HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqL0FWDuwzMDc0Nz
Cache-Control: no-cache
```

### Response

```
[
  {
    "sbj": "af583e70-fe39-407d-a87e-06cd0ec1830c",
    "ua": {
      "broName": "Chrome",
      "broVer": "109",
      "deviceType": "pc",
      "raw": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) ...",
      "osName": "macOS",
      "osVer": "10.15.7"
    },
    "ts": 1637250238015,
    "cAthM": "Basic",
    "ipCt": "Москва",
    "ipRad": 20,
    "cId": "test_app",
    "ip": 1406987879,
    "obj": "af583e70-fe39-407d-a87e-06cd0ec1830c",
    "ipSt": "Москва",
    "lpId": "test_app",
    "pid": "ddeebaba-2dc3-41bb-b539-7f0e472414a3",
    "ipLat": 55.7483,
    "prms": {
      "used_login": "test@yandex.ru",
      "auth_methods": "password",
      "authnDone": "true",
      "id_store": "389-ds"
    },
    "type": "login",
    "ipCtr": "Россия",
    "proc": "profile",
    "ipLng": 37.6171,
    "sid": "54914ac3-0d39-40d3-9617-92e0e7fe07ab"
  }
]
```

## Known devices and sessions

### List of known devices

Method GET `https://login.company.com/blitz/api/v3/users/{subjectId}/uas`

Getting a list of the user's devices.

Required permissions: `blitz_api_uapps` or `blitz_api_sys_uapps`.

Returns JSON containing a list of the user's devices.

### Example

#### Request

```
GET /blitz/api/v3/users/af583e70-fe39-407d-a87e-06cd0ec1830c/uas HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqLOFWDuwzMDc0Nz
Cache-Control: no-cache
```

#### Response

```
[
  {
    "name": "Chrome 96",
    "lastUsed": 1637249978,
    "tp": "Browser",
    "os": "macOS 10.15.7",
    "newlyCreated": false,
    "deviceType": "pc",
    "latestIp": "172.25.0.1",
    "subjectId": "af583e70-fe39-407d-a87e-06cd0ec1830c",
    "id": "SHA256_Z0x284K3qv313WViRuPfv5rglhDuYqSn4ztdxVKMBec",
    "trusted": false,
    "cls": true,
    "deviceId": "738f5ce91f912ddd4a0cc5fef9a9e8c63",
    "device": "PC"
  }
]
```

### Deleting a device from the list

Method DELETE `https://login.company.com/blitz/api/v3/users/{subjectId}/uas/{id}`

Deleting a device from the list of stored ones. As the `id`, you need to pass *received* (page 389) device ID.

Required permissions: `blitz_api_uapps_chg` or `blitz_api_sys_uapps_chg`.

Headers In user mode, headers with the user's IP address and `User-Agent` must be passed.

## Example

Listing 70: Request

```
DELETE /blitz/api/v3/users/d2580c98-e584-4aad-a591-97a8cf45cd2a/uas/SHA256_
↪Z0x284K3qv313WViRuPfv5rglhDuYqSn4ztdxVKMBec HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqLOFDuwzMDc0Nz
X-Forwarded-For: 200.200.100.100
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_5)
```

## Resetting user sessions

Method `POST https://login.company.com/blitz/api/v3/users/{subjectId}/sessions/reset`

Resetting user sessions.

Required permissions: `blitz_api_usec_chg` or `blitz_api_sys_usec_chg`.

Headers

- In user mode, headers with the user's IP address and `User-Agent` must be passed.
- If the user's logout from the current device/browser is undesirable, you need to transfer the `IB-CI-UA-ID` header from the application with the identifier of the current device in order to save the session on it.

---

**Tip:** The ID of the user's current device can be obtained from [маркера идентификации](#) (page 309).

---

Returns If the call is successful, the code is HTTP 204 No Content.

**Attention:** Resetting sessions will invalidate previously received access tokens and refresh tokens of the current user.

## Request examples

Listing 71: User mode

```
POST /blitz/api/v3/users/c574a512-3704-4576-bc3a-3fe28b636e85/sessions/reset HTTP/
↪1.1
Content-Type: application/json
X-Forwarded-For: 200.200.100.100
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_5)...
Authorization: Bearer wzb...Tw
IB-CI-UA-ID: {SHA256}rVWFmwgRKWeW_flH4CA4yuW7OhKZ32Da94m0kzwWsVs
```

Listing 72: System service call mode

```
POST /blitz/api/v3/users/c574a512-3704-4576-bc3a-3fe28b636e85/sessions/reset HTTP/
↪1.1
Content-Type: application/json
Authorization: Bearer qwa...Ez
```

## Security questions

### Checking for a question

Method GET `https://login.company.com/blitz/api/v3/users/{subjectId}/secQsn`

Checking whether the user has a security question.

Required permissions: `blitz_api_usec` or `blitz_api_sys_usec`.

Returns

- If the security question is asked - the text of the security question.
- If the security question is not asked - 404 Not Found.

### Example

#### Request

```
GET /blitz/api/v3/users/d2580c98-e584-4aad-a591-97a8cf45cd2a/secQsn HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqLOFWDuwzMDc0Nz
Cache-Control: no-cache
```

#### Response

```
{
  "question": "Как звали вашего первого питомца"
}
```

### Checking the answer

Method POST `https://login.company.com/blitz/api/v3/users/{subjectId}/secQsn/check`

Checking the correctness of the answer to the security question.

Required permissions: `blitz_api_usec` or `blitz_api_sys_usec`.

Request body A security question (`question`) and the answer to it (`answer`).

Returns

- In case of successful verification of the question and response - 204 No Content.
- Otherwise - 400 Bad request.

### Example

#### Request

```
POST /blitz/api/v3/users/d2580c98-e584-4aad-a591-97a8cf45cd2a/secQsn/check HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqLOFWDuwzMDc0Nz
Content-Type: application/json
X-Forwarded-For: 200.200.100.100
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_5)...
```

(continues on next page)

(continued from previous page)

```
{
  "question": "Как звали вашего первого питомца",
  "answer": "Тигр"
}
```

**Error**

Listing 73: The security question did not match

```
{
  "type": "process_error",
  "error": "wrong_security_answer",
  "desc": "security question not match"
}
```

Listing 74: The answer to the security question did not match

```
{
  "type": "process_error",
  "error": "wrong_security_answer",
  "desc": "security answer not match"
}
```

Listing 75: The user's security question is not set

```
{
  "type": "process_error",
  "error": "wrong_security_answer",
  "desc": "security question not found"
}
```

**Setting or changing a question**

Method POST <https://login.company.com/blitz/api/v3/users/{subjectId}/secQsn>  
Setting or changing the user's security question.

Required permissions: `blitz_api_sys_usec_chg` or `blitz_api_sys_usec_chg`.

Request body A security question (`question`) and the answer to it (`answer`).

Returns In case of successful setting of the security question - 204 No Content.

Listing 76: Request example

```
POST /blitz/api/v3/users/d2580c98-e584-4aad-a591-97a8cf45cd2a/secQsn HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqL0FWDuwzMDc0Nz
Content-Type: application/json
X-Forwarded-For: 200.200.100.100
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_5)...
```

```
{
  "question": "Как звали вашего первого питомца",
  "answer": "Тигр"
}
```

## Deleting a question

Method DELETE `https://login.company.com/blitz/api/v3/users/{subjectId}/secQsn`

Deleting the security question from the user's account.

Required permissions: `blitz_api_usec_chg` or ``blitz_api_sys_usec_chg`.

Returns If successful - 204 No Content.

### Listing 77: Request example

```
DELETE /blitz/api/v3/users/d2580c98-e584-4aad-a591-97a8cf45cd2a/secQsn HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqLOFWDuwzMDc0Nz
X-Forwarded-For: 200.200.100.100
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_5)...
```

## Permissions issued by the user

### List of permissions

Method GET `https://login.company.com/blitz/api/v3/users/{subjectId}/acls`

Getting a list of permissions issued by the user.

Required permissions: `blitz_api_usec` or `blitz_api_sys_usec`.

Returns JSON containing a list of permissions granted by the user.

### Example

#### Request

```
GET /blitz/api/v3/users/d2580c98-e584-4aad-a591-97a8cf45cd2a/acls HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqLOFWDuwzMDc0Nz
Cache-Control: no-cache
```

#### Response

```
[
  {
    "id": "d2580c98 e584 4aad a591 97a8cf45cd2a_app1",
    "updated": 1552896932780,
    "client_id": "app1",
    "scopes": [
      "openid",
      "profile",
    ]
  }
]
```

## Revocation of permission

**Method** DELETE `https://login.company.com/blitz/api/v3/users/{subjectId}/acIs/{acl_id}`

Revocation of the issued permission.

**Required permissions:** `blitz_api_usec_chg` or `blitz_api_sys_usec_chg`.

**URL parameters** The *received* (page 393) identifier (`id`) of the permission is passed as the “`acl_id`”.

**Headers** In user mode, headers with the user’s IP address and `User-Agent` must be passed.

## Example

Listing 78: Request

```
DELETE /blitz/api/v3/users/d25..2a/acIs/d25..2a_app1 HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqLOFWDuwzMDc0Nz
```

## Mobile apps

### List of mobile apps

**Method** GET `https://login.company.com/blitz/api/v3/users/{subjectId}/apps`

Getting a list of linked mobile apps.

**Required permissions:** `blitz_api_uapps` or `blitz_api_sys_uapps`.

**Returns** JSON, containing a list of linked mobile apps.

## Example

### Request

```
GET /blitz/api/v3/users/d2580c98-e584-4aad-a591-97a8cf45cd2a/apps HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqLOFWDuwzMDc0Nz
Cache-Control: no-cache
```

## Response

```
[
  {
    "id": "dyn~test_app~afae0cab-2649-482d-9832-5f73816afb59",
    "name": {
      "_default_": "Тестовое приложение (test_app)"
    },
    "availableScopes": [
      "openid",
      "profile"
    ],
    "softwareId": "test_app"
  }
]
```

### Unlinking from a mobile app account

DELETE `https://login.company.com/blitz/api/v3/users/{subjectId}/apps/{app_id}`

Revocation of the issued permission.

Required permissions: `blitz_api_uapps_chg` or `blitz_api_sys_uapps_chg`.

URL parameters The *received* (page 394) identifier (`id`) of the application linking is passed as the `app_id`.

Headers In user mode, headers with the user's IP address and `User-Agent` must be passed.

#### Example

Listing 79: Request

```
DELETE /blitz/api/v3/users/d2580c98-e584-4aad-a591-97a8cf45cd2a/apps/d2580c98-e584-
↪4aad-a591-97a8cf45cd2a_app1 HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqLOFWDuwzMDc0Nz
X-Forwarded-For: 200.200.100.100
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_5)...
```

### Deleting an account

Method DELETE `https://login.company.com/blitz/api/v2/users/{subjectId}?instanceId={instanceId}`

Deleting the user account.

The `subjectId` contains the identifier of the account to be deleted, and the `instanceId` parameter contains a link to the account to be deleted. To find out the value of `instanceId` for the user, you must first call the *GET service for obtaining attributes* (page 357) of the user.

#### Example

Listing 80: Request

```
DELETE /blitz/api/v2/users/d..2a?instanceId=M..U HTTP/1.1
Authorization: Basic YXBwX2lkOmFwcF9zZWNyZXQ=
```

## 3.4.3 User groups

**Attention:** To call services, the system must obtain an access token to *system permission* (page 340) `blitz_groups` and include it in all called services.

Groups in Blitz Identity Provider are described by the following attributes:

- `id` is the ID of the group in Blitz Identity Provider;
- `name` is the name of the user group.



### Getting group attributes by id

Method GET `https://login.company.com/blitz/api/v2/grps/{id}`

Getting the attributes of the group, if the `id` of the group is known.

URL parameters

- `profile` is the name of the profile of user groups (for example, `orgs`);
- `expand` is the value `true`, indicating that it is necessary to return all the attributes of the group.

### Example

#### Request

```
GET /blitz/api/v2/grps/14339e8e-a665-4556-92f1-5c348eff6696?profile=orgs HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqLOFWDuwzMDc0Nz
Cache-Control: no-cache
```

#### Response

```
{
  "instanceId": "Mzg...nU",
  "id": "14339e8e-a665-4556-92f1-5c348eff6696",
  "OGRN": "1234567890329",
  "INN": "7743151614",
  "name": "ООО Тестовая компания",
  "profile": "orgs"
}
```

### Search for a group by attribute

Method GET `https://login.company.com/blitz/api/v2/grps`

Search for a group by attribute and getting all its attributes if the `id` of the group is unknown.

URL parameters

- `profile` is the name of the user groups profile;
- `rql` is a search query for group attributes in the format [Resource Query Language<sup>87</sup>](#) (RQL).

Operations:

- `and` - simultaneous execution of search conditions;
- `or` - alternative fulfillment of search conditions (for example, search by different attributes);
- `eq` - checking the equality condition;
- `limit` - a limit on the number of records to be returned.

- `expand` (optional parameter):
  - `true`: include group attributes in the received response;
  - `false`: return only the IDs of the found groups.

Returns JSON, containing a list of groups that meet the specified search conditions, indicating their identifier (`id`), as well as the values of the other attributes of the groups (in the case of `expand=true`).

<sup>87</sup> <https://github.com/kriszyp/rql>

## Example

### Request

Listing 81: Search for a group by PSRN or TIN

```
GET /blitz/api/v2/grps?profile=orgs&expand=true&rql=or(eq(OGRN,
↪string:1230123456789),eq(INN,string:7743151614)) HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqLOFWDuwzMDc0Nz
Cache-Control: no-cache
```

### Response

```
[
  {
    "instanceId": "Mzg5L...nU",
    "id": "14339e8e-a665-4556-92f1-5c348eff6696",
    "OGRN": "1234567890329",
    "INN": "7743151614",
    "name": "ООО Тестовая компания",
    "profile": "orgs"
  }
]
```

## Creating a group

Method POST <https://login.company.com/blitz/api/v2/grps>

Creating a user group.

Request body

- `profile` is the name of the user groups profile;
- `id` is the unique identifier of the group;
- the rest of the group's attributes and their values.

## Example

### Request

```
POST /blitz/api/v2/grps HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqLOFWDuwzMDc0Nz
Content-Type: application/json

{
  "id": "95339e8e-a665-4556-92f1-5c348eff6696",
  "OGRN": "9876543210321",
  "INN": "5012345678",
  "name": "ООО Тестовая компания 2",
  "profile": "orgs"
}
```

## Response

```
{
  "instanceId": "b3Jnc...dQ",
  "name": "ООО Тестовая компания 2",
  "OGRN": "9876543210321",
  "id": "95339e8e-a665-4556-92f1-5c348eff6696",
  "profile": "orgs",
  "INN": "5012345678"
}
```

## Changing group attributes

Method POST <https://login.company.com/blitz/api/v2/grps/{id}?profile=orgs>

Changing group attributes.

Request body New set of attributes:

- `profile` – the name of the group profile (must be passed both as part of the URL and in the request body);
- `id` – group identifier;
- the rest of the group's attributes and their values.

## Example

### Request

```
POST /blitz/api/v2/grps/5f7b0580-cd2e-4146-8fc5-6eb5a95c7b42?profile=orgs HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqLOFWDuwzMDc0Nz
Content-Type: application/json
```

```
{
  "id": "5f7b0580-cd2e-4146-8fc5-6eb5a95c7b42",
  "OGRN": "1147746651733",
  "INN": "7715434658",
  "name": "Новое название",
  "profile": "orgs"
}
```

### Response

```
{
  "instanceId": "Mzg5L...nU",
  "id": "5f7b0580-cd2e-4146-8fc5-6eb5a95c7b42",
  "OGRN": "1147746651733",
  "INN": "7715434658",
  "name": "Новое название",
  "profile": "orgs"
}
```

## Error

Listing 82: The organization does not exist

```
{
  "errors": [
    {
      "code": "group_not_found",
      "desc": "Group with '95339e8e-...97' id not found in '389-ds' LDAP group_
↪store",
      "params": {}
    }
  ]
}
```

## Deleting a group

Method DELETE <https://login.company.com/blitz/api/v2/grps/{id}?profile=orgs>

Deleting a group.

## Example

Listing 83: Request

```
DELETE /blitz/api/v2/grps/5f7b0580-cd2e-4146-8fc5-6eb5a95c7b42?profile=orgs HTTP/1.
↪1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqLOFWDuwzMDc0Nz
```

## Getting a list of users in a group

Method GET <https://login.company.com/blitz/api/v2/grps/{id}/members>

Getting a list of users from a group.

URL parameters

- `profile` is the name of the user groups profile;
- `expand` (optional parameter):
  - `true`: include the user's full name in the received response;
  - `false`: return only user IDs.

## Example

### Request

Listing 84: `expand=false`

```
GET /blitz/api/v2/grps/14339e8e-a665-4556-92f1-5c348eff6696/members?profile=orgs&
↪expand=false HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqLOFWDuwzMDc0Nz
Cache-Control: no-cache
```

Listing 85: expand=true

```
GET /blitz/api/v2/grps/14339e8e-a665-4556-92f1-5c348eff6696/members?profile=orgs&
↳expand=true HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqLOFWDuwzMDc0Nz
Cache-Control: no-cache
```

### Response

Listing 86: expand=false

```
[
  {
    "instanceId": "Mzg5L...J1",
    "subjectId": "d434b7d4-9816-460a-83aa-0a994226cbe7"
  },
  {
    "instanceId": "Mzg5L...J1",
    "subjectId": "2cafa5f4-bc84-4f6f-91aa-080da47975f0"
  }
]
```

Listing 87: expand=true

```
[
  {
    "instanceId": "Mzg5L...J1",
    "family_name": "Иванов",
    "middle_name": "Иванович",
    "given_name": "Иван",
    "subjectId": "d434b7d4-9816-460a-83aa-0a994226cbe7"
  },
  {
    "instanceId": "Mzg5L...J1",
    "family_name": "Сергеев",
    "middle_name": "Сергеевич",
    "given_name": "Сергей",
    "subjectId": "2cafa5f4-bc84-4f6f-91aa-080da47975f0"
  }
]
```

## Adding users

Method `POST https://login.company.com/blitz/api/v2/grps/{id}/members/add?profile=orgs`

Adding users to a group.

Request body A list of users to be added to the group with their IDs (`sub`) in the `subjectId` attribute.

## Request

```
POST /blitz/api/v2/grps/5f7b0580-cd2e-4146-8fc5-6eb5a95c7b42/members/add?
↪profile=orgs HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqLOFWDuwzMDc0Nz
Content-Type: application/json

[
  {
    "subjectId": "45ff69f2-6c40-418f-a21d-cbe6f07b88c9"
  },
  {
    "subjectId": "cc8c4589-b2f8-40b8-b351-36d643808943"
  }
]
```

## Response

```
[
  {
    "instanceId": "Mzg5L...J1",
    "storeId": "tam",
    "subjectId": "45ff69f2-6c40-418f-a21d-cbe6f07b88c9"
  },
  {
    "instanceId": "Nzg5L...J1",
    "storeId": "tam",
    "subjectId": "cc8c4589-b2f8-40b8-b351-36d643808943"
  }
]
```

## Error

Listing 88: Attempt to add a non-existent user

```
{
  "errors": [
    {
      "code": "user_not_found",
      "desc": "User with subjectId 'd2580c98-e584-4aad-a591-97a8cf45cd2q' ↪
↪not found",
      "params": {}
    }
  ]
}
```

Listing 89: An attempt to add a user who is already in the group

```
{
  "errors": [
    {
      "code": "some_members_already_in_group",
      "desc": "Some of adding members are already included in group",
      "params": {}
    }
  ]
}
```

### Removing users

Method POST `https://login.company.com/blitz/api/v2/grps/{id}/members/rm?profile=orgs`

Removing users from the group.

Request body A list of trusted persons excluded from the organization, indicating their identifiers (`sub`) in the `subjectId` attribute.

### Request

```
POST /blitz/api/v2/grps/5f7b0580-cd2e-4146-8fc5-6eb5a95c7b42/members/rm?
↳profile=orgs HTTP/1.1
Authorization: Bearer cNwIXatB0wk5ZH00xG5kxuuLubesWcb_yPPqLOFWDuwzMDc0Nz
Content-Type: application/json
```

```
[
  {
    "subjectId": "d2580c98-e584-4aad-a591-97a8cf45cd2a"
  }
]
```

### Response

```
[
  {
    "instanceId": "Mzg5L...J1",
    "storeId": "389-ds",
    "subjectId": "d2580c98-e584-4aad-a591-97a8cf45cd2a"
  }
]
```

## Error

Listing 90: An attempt to delete a user from the group who is no longer in it

```

{
  "errors": [
    {
      "code": "some_members_not_in_group",
      "desc": "Some of removing members are not included in group",
      "params": {}
    }
  ]
}

```

Listing 91: Attempt to delete a non-existent user

```

{
  "errors": [
    {
      "code": "user_not_found",
      "desc": "User with subjectId 'd2580c98-e584-4aad-a591-97a8cf45cd2b' ↵
↵not found",
      "params": {}
    }
  ]
}

```

### 3.4.4 Access rights

**Attention:** To make requests for viewing, assigning, revoking access rights, the application must receive an access token with the system permission `blitz_rights_full_access`.

**Tip:** To view the access rights of a user where he is a subject, you can also use an access token with the user permission `blitz_user_rights`.

The access right is assigned from the access subject to the access object.

Access subjects:

- users,
- applications (prefix `its`).

Access objects:

- users,
- user groups (prefix `grps`),
- applications (prefix `its`).



### List of user rights

Method GET <https://login.company.com/blitz/api/v3/rights/of/<sub>>

Obtaining access rights by the access subject who is the user.

### Examples

#### Request

```
GET /blitz/api/v3/rights/of/BIP-1SEQ41A HTTP/1.1
Authorization: Bearer cNwIX...Nz
```

#### Response

Listing 92: The user BIP-1SEQ41A has the right ORG\_ADMIN to the user group 1147746651733, the right APP\_ADMIN to the application test\_app2, the right change\_password to the user account BIP-3SGR7TA

```
{
  "grps|1147746651733|orgs": {
    "ORG_ADMIN": [
      "set_from_api",
      "another_one_tag"
    ]
  },
  "its|test_app2": {
    "APP_ADMIN": [
      "set_from_api"
    ]
  },
  "BIP-3SGR7TA": {
    "change_password": [
      "parent"
    ]
  }
}
```

### List of application rights

Method GET [https://login.company.com/blitz/api/v3/rights/of/its/<app\\_id>](https://login.company.com/blitz/api/v3/rights/of/its/<app_id>)

Obtaining access rights by the access subject that is the application.

## Examples

### Request

```
GET /blitz/api/v3/rights/of/its/test_app HTTP/1.1
Authorization: Bearer cNwIX...Nz
```

### Response

Listing 93: The application `test_app` has the right `SYS_MON` to the application `test_app2`, the right `change_password` to the user account `BIP-3SGR7TA`, the right `ORG_ADMIN` to the user group `1147746651733`

```
{
  "its|test_app2": {
    "SYS_MON": [
      "set_from_api"
    ]
  },
  "BIP-3SGR7TA": {
    "change_password": [
      "set_from_api"
    ]
  },
  "grps|1147746651733|orgs": {
    "ORG_ADMIN": [
      "set_from_api"
    ]
  }
}
```

### Rights in relation to the user

Method GET <https://login.company.com/blitz/api/v3/rights/on/<sub>>

Obtaining access rights for an access object that is a user.

## Examples

### Request

```
GET /blitz/api/v3/rights/on/BIP-3SGR7TA HTTP/1.1
Authorization: Bearer cNwIX...Nz
```

## Response

Listing 94: The user BIP-1SEQ41A and the application test\_app have the right change\_password for the account BIP-3SGR7TA

```
{
  "BIP 1SEQ41A": [
    "change_password"
  ],
  "its|test_app": [
    "change_password"
  ]
}
```

## Rights in relation to a group of users

Method GET https://login.company.com/blitz/api/v3/rights/on/grps/<grp\_id>?objectExt=<profile>

Obtaining access rights for an access object that is a group.

## Examples

### Request

```
GET /blitz/api/v3/rights/on/grps/1147746651733?objectExt=orgs HTTP/1.1
Authorization: Bearer cNwIX..Nz
```

## Response

Listing 95: The user BIP-1SEQ41A, and the application test\_app has the right *ORG\_ADMIN* for the account of the group 1147746651733 from the profile *orgs*

```
{
  "BIP 1SEQ41A": [
    "ORG_ADMIN"
  ],
  "its|test_app": [
    "ORG_ADMIN"
  ]
}
```

## Rights in relation to the application

Method GET https://login.company.com/blitz/api/v3/rights/on/its/<app\_id>

Obtaining access rights for an access object that is an application.

## Examples

### Request

```
GET /blitz/api/v3/rights/on/its/test_app2 HTTP/1.1
Authorization: Bearer cNwIX...Nz
```

### Response

Listing 96: The user BIP-1SEQ41A has the *APP\_ADMIN* right to the *test\_app2* application account, and the *test\_app* application has the *SYS\_MON* right

```
{
  "BIP 1SEQ41A": [
    "APP_ADMIN"
  ],
  "its|test_app": [
    "SYS_MON"
  ]
}
```

### Error

Listing 97: If the access token is expired, the service will return the error HTTP 401 Unauthorized and JSON

```
{
  "type": "security_error",
  "error": "bad_access_token",
  "desc": "expired_access_token"
}
```

## Assignment of rights

Method PUT <https://login.company.com/blitz/api/v3/rights>

Assigning access rights.

Request body

- `subject` is the identifier of the subject to whom the right is assigned (user or application identifier);
- `subjectType` is the type of the subject. The parameter is specified only if the right is assigned to the application. In this case, the value `its` is used;
- `object` is the identifier of the object to which the right is assigned (the identifier of a user, user group, or application);
- `objectType` is the type of the object. The parameter is specified only if the right is assigned to a user group (value `grps`) or to an application (value `its`);
- `rights` is an array with a list of assigned rights to the subject on the object;
- `tags` is an array with a list of tags of assigned rights.

Returns

- In case of successful assignment of access rights - HTTP 204 No Content.

- If the access token is expired - HTTP 401 Unauthorized.
- If the subject or object does not exist - HTTP 400 Bad Request

## Examples

### Request

Listing 98: Assigning access rights to a user to another user

```
PUT /blitz/api/v3/rights HTTP/1.1
Authorization: Bearer cNwIXNz
Content-Type: application/json

{
  "subject": "BIP-1SEQ41A",
  "object": "BIP-3SGR7TA",
  "rights": ["change_password"],
  "tags": ["set_from_api"]
}
```

Listing 99: Assigning user access rights to a group

```
PUT /blitz/api/v3/rights HTTP/1.1
Authorization: Bearer cNwIXNz
Content-Type: application/json

{
  "subject": "BIP-1SEQ41A",
  "object": "1147746651733",
  "objectType": "grps",
  "rights": ["ORG_ADMIN"],
  "tags": ["set_from_api"]
}
```

Listing 100: Assigning user access rights to an application

```
PUT /blitz/api/v3/rights HTTP/1.1
Authorization: Bearer cNwIXNz
Content-Type: application/json

{
  "subject": "BIP-1SEQ41A",
  "object": "test_app2",
  "objectType": "its",
  "rights": ["APP_ADMIN"],
  "tags": ["set_from_api"]
}
```

Listing 101: Assigning access rights to an application to a user

```
PUT /blitz/api/v3/rights HTTP/1.1
Authorization: Bearer cNwIXNz
Content-Type: application/json

{
  "subject": "test_app",
  "subjectType": "its",
```

(continues on next page)

(continued from previous page)

```

"object": "BIP-3SGR7TA",
"rights": ["change_password"],
"tags": ["set_from_api"]
}

```

Listing 102: Assigning access rights to an application for a group

```

PUT /blitz/api/v3/rights HTTP/1.1
Authorization: Bearer cNwIXNz
Content-Type: application/json

{
  "subject": "test_app",
  "subjectType": "its",
  "object": "1147746651733",
  "objectType": "grps",
  "rights": ["ORG_ADMIN"],
  "tags": ["set_from_api"]
}

```

Listing 103: Assigning access rights to an application to another application

```

PUT /blitz/api/v3/rights HTTP/1.1
Authorization: Bearer cNwIXNz
Content-Type: application/json

{
  "subject": "test_app",
  "subjectType": "its",
  "object": "test_app2",
  "objectType": "its",
  "rights": ["SYS_MON"],
  "tags": ["set_from_api"]
}

```

**Error**

Listing 104: The access token has expired

```

{
  "type": "security_error",
  "error": "bad_access_token",
  "desc": "expired_access_token"
}

```

Listing 105: The assigned right does not exist

```

{
  "type": "process_error",
  "error": "unknown_right",
  "desc": "The specified right is unknown",
  "params": {
    "right": "change_password1"
  }
}

```

Listing 106: The user specified as the subject or object does not exist

```
{
  "type": "process_error",
  "error": "unknown_user",
  "desc": "The specified user is unknown",
  "params": {
    "userId": "ivanov1"
  }
}
```

Listing 107: The group specified as an object does not exist

```
{
  "type": "process_error",
  "error": "unknown_group",
  "desc": "The specified group is unknown",
  "params": {
    "grpId": "1147746651734"
  }
}
```

Listing 108: The specified application subject or object does not exist

```
{
  "type": "process_error",
  "error": "unknown_rp",
  "desc": "The specified relying party is unknown",
  "params": {
    "rpId": "test_app3"
  }
}
```

## Revocation of rights

Method DELETE <https://login.company.com/blitz/api/v3/rights>

Revocation of access rights.

Request body

- `subject` is the identifier of the subject whose right is being revoked (user or application ID);
- `subjectType` is the type of the subject. The parameter is specified only in case of revocation of the application's rights. In this case, the value `its` is used;
- `object` is the identifier of the object to which the right is being revoked (the identifier of a user, user group, or application);
- `objectType` is the type of the object. The parameter is specified only in case of revocation of the right to a user group (value `grps`) or to an application (value `its`);
- `rights` is an array with a list of revoked rights of the subject to the object;
- `tags` is an array with a list of tags of revoked rights.

**Warning:** If an access right has been assigned to an access subject for an access object with multiple tags, then all tags must also be specified to revoke the access right. If revocation of access rights is not called with full indication of tags, then only the revoked tags will be deleted during revocation,

and the access right of the access subject to the access object will remain as long as at least one of the tags remains.

#### Returns

- In case of successful revocation of the access right, the service will return HTTP 204 No Content.
- If the access token is expired - HTTP 401 Unauthorized.
- If the revoked right, subject or object does not exist - HTTP 400 Bad Request

#### Examples

##### Request

Listing 109: Revoking a user's access rights to another user

```
DELETE /blitz/api/v3/rights HTTP/1.1
Authorization: Bearer cNwIXNz
Content-Type: application/json

{
  "subject": "BIP-1SEQ41A",
  "object": "BIP-3SGR7TA",
  "rights": ["change_password"],
  "tags": ["set_from_api"]
}
```

Listing 110: Revoking a user's access rights to a group

```
DELETE /blitz/api/v3/rights HTTP/1.1
Authorization: Bearer cNwIXNz
Content-Type: application/json

{
  "subject": "BIP-1SEQ41A",
  "object": "1147746651733",
  "objectType": "grps",
  "rights": ["ORG_ADMIN"],
  "tags": ["set_from_api"]
}
```

Listing 111: Revoking the user's access rights to the application

```
DELETE /blitz/api/v3/rights HTTP/1.1
Authorization: Bearer cNwIXNz
Content-Type: application/json

{
  "subject": "BIP-1SEQ41A",
  "object": "test_app2",
  "objectType": "its",
  "rights": ["APP_ADMIN"],
  "tags": ["set_from_api"]
}
```



Listing 112: Revoking the application's access rights to the user

```
DELETE /blitz/api/v3/rights HTTP/1.1
Authorization: Bearer cNwIXNz
Content-Type: application/json

{
  "subject": "test_app",
  "subjectType": "its",
  "object": "BIP-3SGR7TA",
  "rights": ["change_password"],
  "tags": ["set_from_api"]
}
```

Listing 113: Revoking the application's access rights to the group

```
DELETE /blitz/api/v3/rights HTTP/1.1
Authorization: Bearer cNwIXNz
Content-Type: application/json

{
  "subject": "test_app",
  "subjectType": "its",
  "object": "1147746651733",
  "objectType": "grps",
  "rights": ["ORG_ADMIN"],
  "tags": ["set_from_api"]
}
```

Listing 114: Revoking the application's access rights to another application

```
DELETE /blitz/api/v3/rights HTTP/1.1
Authorization: Bearer cNwIXNz
Content-Type: application/json

{
  "subject": "test_app",
  "subjectType": "its",
  "object": "test_app2",
  "objectType": "its",
  "rights": ["SYS_MON"],
  "tags": ["set_from_api"]
}
```

## Response

Listing 115: The access token has expired

```
{
  "type": "security_error",
  "error": "bad_access_token",
  "desc": "expired_access_token"
}
```

Listing 116: The revoked right does not exist

```
{
  "type": "process_error",
  "error": "unknown_right",
  "desc": "The specified right is unknown",
  "params": {
    "right": "change_password1"
  }
}
```

Listing 117: The user specified as the subject or object does not exist

```
{
  "type": "process_error",
  "error": "unknown_user",
  "desc": "The specified user is unknown",
  "params": {
    "userId": "ivanov1"
  }
}
```

Listing 118: The group specified as an object does not exist

```
{
  "type": "process_error",
  "error": "unknown_group",
  "desc": "The specified group is unknown",
  "params": {
    "grpId": "1147746651734"
  }
}
```

Listing 119: The specified application subject or object does not exist

```
{
  "type": "process_error",
  "error": "unknown_rp",
  "desc": "The specified relying party is unknown",
  "params": {
    "rpId": "test_app3"
  }
}
```

### The rights of the master user in relation to the slave

Method POST <https://login.company.com/blitz/api/v2/users/rights/change>

Assigning and revoking the rights of the master user in relation to the slave user.

**Attention:** A revocation request can be executed by an application not only using a user access token obtained for permission named `blitz_user_rights`, but also using a system access token obtained for permission named `blitz_rm_rights`. In this case, the revocation request may include the “subject” of any users (to revoke a user’s rights, it will not be necessary for this particular user to log in and receive an access token – the system can revoke the rights of any user).

**Headers** A header with a permission access token named `blitz_user_rights` received by the lead user account should be added to the request.

**Request body**

### Assignment of rights

A completed `update` block with a list of rights that should be added as a result of the operation.

Each right is described by the parameters:

- `subject` is the identifier (`sub`) of the lead user account;
- `object` is the identifier (`sub`) of the slave user account;
- `rights` is a list of rights in the form of an array that the account of the lead user receives in relation to the account of the slave user. For example, for the right to change the account password, the `change_password` right must be specified (`password change`);
- `tags` is a list of tags indicating the reasons for which this user received rights.

### Revocation of rights

A completed `delete` block with a list of rights that should be revoked as a result of the operation.

Each right is described by the parameters:

- `subject` is the identifier (`sub`) of the lead user account;
- `object` is the identifier (`sub`) of the slave user account;
- `rights` is a list of rights in the form of an array that are revoked from the master account in relation to the slave account;
- `tags` is a list of tags indicating the reasons for which this user received rights.

If the rights are not assigned or revoked during the execution of the request, then either an empty `update` block or an empty `delete` block must be present in the request body, respectively. Several assignable/revocable rights can be specified in a single request, but only the user to whom the access token used to call the service was received must be specified as the subject (`subject`).

## Examples

### Request

Listing 120: Assignment of rights

```
POST /blitz/api/v2/users/rights/change HTTP/1.1
Authorization: Bearer cNwIXTg
Content-Type: application/json

{
  "update": [
    {
      "subject": "6561d0d9-5583-4bb5-a681-b591358e5fcd",
      "object": "5cffd68f-2cb8-4f7a-b0f3-9fa69a1fbbcd",
      "rights": [
        "change_password"
      ],
      "tags": [
        "parent"
      ]
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```

    },
    {
      "subject": "6561d0d9-5583-4bb5-a681-b591358e5fcd",
      "object": "b855957d-bf24-48d4-bb63-cce4f5064590d",
      "rights": [
        "change_password"
      ],
      "tags": [
        "parent"
      ]
    }
  ],
  "delete": [
  ]
}

```

Listing 121: Revocation of rights

```

POST /blitz/api/v2/users/rights/change HTTP/1.1
Authorization: Bearer cNwIXTg
Content-Type: application/json

{
  "update": [
  ],
  "delete": [
    {
      "subject": "b855957d-bf24-48d4-bb63-cce4f5064590d",
      "object": "5cffd68f-2cb8-4f7a-b0f3-9fa69a1fbbcd",
      "rights": [
        "change_password"
      ],
      "tags": [
        "parent"
      ]
    }
  ]
}

```

**Error**

Listing 122: In case of an error, the request is rejected in its entirety and a list of errors is returned

```

{
  "errors" : [
    {
      "code" : "validation_error",
      "params" : {},
      "desc" : "(For subject 'dea75b73-a2ba-4b60-a41c-bb640968826b') Incorrect_
↪right '' to object '5cffd68f-2cb8-4f7a-b0f3-9fa69a1fbbcd'"
    },
    {
      "params" : {},
      "code" : "validation_error",
      "desc" : "(For subject 'dea75b73-a2ba-4b60-a41c-bb640968826b') Incorrect_
↪tag '' for right 'write' to object '5cffd68f-2cb8-4f7a-b0f3-9fa69a1fbbcd'"
    },
  ],
}

```

(continues on next page)

(continued from previous page)

```

    {
      "desc" : "(For subject 'dea75b73-a2ba-4b60-a41c-bb640968826b') Incorrect_
↵object  ",
      "code" : "validation_error",
      "params" : {}
    },
    {
      "desc" : "Incorrect subject ''",
      "code" : "validation_error",
      "params" : {}
    }
  ]
}

```

## 3.5 Advanced features

### 3.5.1 Additional authentication method

Blitz Identity Provider allows you to connect your own developed authentication method. To do this, the system acting as a provider of such an authentication method must:

- provide an authentication request handler;
- send the authentication result to Blitz Identity Provider;
- provide the authentication method applicability verification method Optional.

In Blitz Identity Provider, the developed authentication method must be registered as an [external authentication method](#) (page 108).

#### Request handler service

The interaction of Blitz Identity Provider with the authentication request handler service is performed as follows:

1. The handler service is a URL for receiving HTTP requests from Blitz Identity Provider. When requesting authentication, Blitz Identity Provider will make a POST request to this address.

In the request body, Blitz Identity Provider will transmit the following data in JSON format:

- request ID (`id`);
- statements characterizing the user (`claims`) are optional, only when called as a second factor;
- the ID of the system that requested the login (`rpId`);
- authentication context identifier (`loginContextId`);
- request data (`request`), which includes headers (`headers`), the user's IP address (`remoteAddress`), the method address (`uri`), a list of cookie (`cookies`) and the user's User Agent (`userAgent`).

Listing 123: Example of the request body

```

{
  "id": "a9692091-4613-41aa-91d2-9a71a3fc2e07",
  "claims": {},
  "rpId": "_blitz_profile",
  "loginContextId": "4502aa51-f28c-4a64-951c-5ab1e77b1294",
  "request": {
    "headers": {},

```

(continues on next page)

(continued from previous page)

```

    "remoteAddress": "172.25.0.1",
    "uri": "/blitz/login/methods2/outside_test",
    "cookies": {},
    "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:80.0)..."
  }
}

```

2. Blitz Identity Provider request must be handled on the side of the external method provider . As a result, the external method must return:

- If authentication is possible - an HTTP response to be executed in the user's browser, which, for example, contains the HTML page code or initiates a browser redirect to the required page of the external method.
- An error if user authentication is impossible.

### Blitz Identity Provider request handling requirements

#### HTTP response

- the response should include a cookie setup (on the shared Blitz Identity Provider and external method domain);
- the cookie name must be pre-registered in Blitz Identity Provider;
- the session ID generated by an external method must be used as the cookie value.

Listing 124: Example of an HTTP response with a redirect and cookie setup

```

HTTP/1.1 302 Found
Location: https://login.company.com/blitz/begin?id=a9692091-4613-41aa-91d2
Set-Cookie: Bmr=YTk2OTIwOTEtNDYxMy00MWFhLTkxZDIOWE3MWEzZmMyZTA3;┘
↳Domain=company.com; path=/blitz; Secure; HttpOnly

```

**Important:** When passing the external method, the provider must verify that the `cookie` value for this request has not been changed.

#### Error

Recommended return codes:

HTTP code	response	Response value	Description of the response
200		OK	Initiating an external method by displaying the page content
302		Found	Initiating an external method through a redirect
400		Bad Request	Required request parameters are missing
500		Internal Server Error	Incoming request handling internal error

## Transmission of the authentication result

After passing the external method, the provider must perform the following actions:

1. The server part of the provider must call Blitz Identity Provider using the POST method at:

```
https://login.company.com/blitz/login/methods/outside/save?methodName=outside\_
→ {name}
```

In this request, `name` is the name of the external method assigned to it in Blitz Identity Provider during registration.

### Request body

#### Successful authentication

If authentication is successful, the request body must specify:

- `request ID (id)`;
- `extSessionId` is the session ID generated by an external method. The ID must match the value passed in the original cookie request;
- `claims` is a list of statements that need to enrich the user's session. The list may be empty;
- `subjectId` is the user ID (only for the first factor; when calling an external method, the user ID cannot be passed as the second factor);
- `loginContextId` is the authentication context ID corresponding to the original request.

Listing 125: Request example

```
POST /blitz/login/methods/outside/save?methodName=outside_test HTTP/1.1
Content-Type: application/json

{
  "id": "426b5139-e4f7-41e6-a206-9503de6f34dd",
  "extSessionId": "YTk2OTIwOTEtNDYxMy00MWFhLTkxZDIOWE3MWEzZmMyZTA3",
  "claims": {},
  "loginContextId": "3ca4d1f0-654a-4665-be98-d105ab6ec35d",
  "subjectId": "2db787c7-6e37-4018-abe9-2bea1011c047"
}
```

### Authentication error

In case of an error, the request body must specify:

- `id` is the request ID;
- `extSessionId` is the session ID generated by an external method. The ID must match the value passed in the original cookie request;
- `error` is error code;
- `msg` is a text description of the error (optional).

Listing 126: Request example

```
POST /blitz/login/methods/outside/save?methodName=outside_test HTTP/1.1
Content-Type: application/json
```

(continues on next page)

(continued from previous page)

```
{
  "id": "426b5139-e4f7-41e6-a206-9503de6f34dd",
  "extSessionId": "YTk2OTIwOTEtNDYxMy00MWFhLTkxZDI0WE3MWEzZmMyZTA3",
  "error": "not_found",
  "msg": "User not found"
}
```

If the authentication results are saved (both successful and unsuccessful), Blitz Identity Provider returns the HTTP 200 OK response.

2. The browser part of the provider should ensure that the user is redirected back to Blitz Identity Provider. To do this, you need to redirect the browser to:

```
https://login.company.com/blitz/login/methods/outside/callback?
↔methodName=outside\_{name}
```

In this request, `name` is the name of the external method assigned to it in Blitz Identity Provider during registration.

### Method verification service

The authentication method applicability verification service is a URL for receiving HTTP requests from Blitz Identity Provider. Prior to the authentication request, Blitz Identity Provider will make a POST request to this address, passing the same data in the body in JSON format as in the authentication request.

As a response, the external method should return JSON with the following attributes:

- request ID (`id`);
- the applicability verification result (`result`), which takes either `true` (the method is applicable) or `false` (the method is not applicable) value;
- the authentication context ID (`loginContextId`) corresponding to the request.

If the service returns `false` as the applicability verification result, then Blitz Identity Provider will not execute the authentication request for this user.

### 3.5.2 Invoking the auxiliary application at the moment of login

At the moment of logging in, Blitz Identity Provider can invoke an auxiliary application that will perform additional operations (for example, show the user an information message or request data updating), after which it will return the user to Blitz Identity Provider for subsequent logging into the target application.

From a technical point of view, the auxiliary application must perform the following actions:

- handle a request to open the auxiliary application,
- return the user to Blitz Identity Provider after handling is completed.



## Request to open the application

A request to invoke the auxiliary application is received as follows:

1. The auxiliary application is accessed by redirecting the user to the link provided by the application. The link will contain the authorization code (`code`) as an option.

Listing 127: Example of a link to initiate a request

```
https://<app_hostname>/?lang=ru&theme=default&code=0Tj...qw
```

2. The application must exchange the authorization code for an access token according to the OAuth 2.0 specification. The access token will be used to obtain the session ID to return the user to Blitz Identity Provider, as well as user data if necessary.

## Example

### Request

```
curl -k -d "grant_type=authorization_code&redirect_uri=https%3A%2F%2Fapp.
↪company.com%2F&client_id=app&client_secret=EW...l0&code=0Tj...qw" -X POST https:/
↪/login.company.com/blitz/oauth/te
```

### Received access token

```
{
  "access_token": "eyJ9.eyJn0.Wa...Pw",
  "token_type": "Bearer",
  "expires_in": 3600,
  "scope": "profile"
}
```

**Important:** The auxiliary application must be pre-registered in Blitz Identity Provider, taking into account the following features:

- a predefined return URL must be specified, which should then be used to receive the token;
- the default permissions (`scope`) must be configured, they determine the amount of data received by the auxiliary application.

## Returning the user to Blitz Identity Provider

The user is returned to Blitz Identity Provider as follows:

1. After completing the necessary actions (for example, showing the user an informational message), the auxiliary application should return the user to Blitz Identity Provider. To do this, you need to decode the received access token, received in JWT format, and extract from it the statement with the user's session (`sessionId`).

Listing 128: Example of the decoded `access_token` body

```
{
  "scope": "blitz_api_user blitz_api_user_chg blitz_api_usec_chg",
  "jti": "kfP...jA",
```

(continues on next page)

(continued from previous page)

```

"client_id": "app",
"exp": 1631026605,
"sessionId": "ce9f3109-ac79-46b4-b277-099ff1aa1ff0",
"iat": 1631023005,
"sub": "8b970179-e141-43b9-b9d5-25997be99261",
"aud": [
  "app"
],
"cruid": "u9th2LzMXZdwb3rRmI3Paw",
"iss": "https://login.company.com/blitz"
}

```

2. After decoding the access token, the auxiliary application must make a POST request to the URL of the authentication completion handler Blitz Identity Provider `/login/pipe/save/<sessionId>`. The request body may contain a set of statements (*claims*) to be added to the user's session, or error information (error).

Listing 129: Request example

```

curl -v --location --request POST 'https://login.company.com/blitz/login/pipe/
↪save/ce9f3109-ac79-46b4-b277-099ff1aa1ff0' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic Z2...ww' \
--data-raw '{"claims":{"org_id":"12345678"}}'

```

3. If successful, Blitz Identity Provider will return HTTP 204 No Content. After receiving it, the auxiliary application should return the user's browser to the address `/login/pipe/callback` so that the user completes logging in to the target application.

Listing 130: Example of a redirect link

```
https://login.company.com/blitz/login/pipe/callback
```

### 3.5.3 Administration API

You can administer Blitz Identity Provider using:

- admin console;
- configuration files;
- administrative REST services.

Administrative REST services in Blitz Identity Provider in the current version allow you to perform the following actions:

- application registration;
- get application settings;
- change application settings;
- delete applications.

Administrative REST services are available at `https://login.company.com/blitz/admin/api/v3/...`

To enable administrative services, settings must first be made on the web server used by Blitz Identity Provider. It is not recommended to publish administrative REST services on the Internet.

An example of the location block in the nginx web server settings to enable the availability of administrative REST services:

```
location /blitz/admin/api {
    proxy_intercept_errors off;
    proxy_pass http://blitz-console/blitz/admin/api;
}
```

Access to administrative REST services is regulated using the permissions (scope) listed in the table:

### Permissions (scope) for administrative REST APIs

No.	Permission	Name	Description
1.	blitz_api_sys_app	Permission to read application settings	To use the service GET /blitz/admin/api/v3/app/{appId}
2.	blitz_api_sys_app	Permission to make changes to application settings	To use the services: PUT /blitz/admin/api/v3/app/{appId} POST /blitz/admin/api/v3/app/{appId} DELETE /blitz/admin/api/v3/app/{appId}

To get an access token for system permission, the application must make a POST request to the URL to receive the token (<https://login.company.com/blitz/oauth/te>). The request must contain the header **Authorization** with the value `Basic {secret}`, where `secret` is `client_id:client_secret` (for example, `app:topsecret`) in Base64 format.

Example of a header:

```
Authorization: Basic YWlzOm...XQ=
```

The request body must contain the following parameters:

- `grant_type` – takes the value `client_credentials`;
- `scope` is the requested system permission.

Request example:

```
POST blitz/oauth/te HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Authorization: Basic ZG5ld...lg

grant_type=client_credentials&scope=blitz_api_sys_app+blitz_api_sys_app_chg
```

In response, the application will receive an access token (`access_token`), its lifetime (`expires_in`) and the token type (`token_type`). Possible errors when calling `/oauth/te` correspond to [RFC 6749](#)<sup>88</sup>.

Example of a response with successful execution of the request:

```
{
  "access_token": "QFiJ9mPgERPuud36mQvD4mfzYolH_CmuddAJ3YKTOI",
  "expires_in": 3600,
  "scope": "blitz_api_sys_app blitz_api_sys_app_chg",
  "token_type": "Bearer"
}
```

<sup>88</sup> <https://tools.ietf.org/html/rfc6749#section-5.2>

It is recommended that the application caches the received access token for repeated use for a time slightly less than the `expires_in` parameter, after which it receives a new access token for updating in the cache.

If the application tries to call the corresponding REST service with an expired access token, it will receive the error HTTP 401 Unauthorized.

### Getting application settings

To get the application settings by its identifier, you need to use the GET method to call the service at `https://login.company.com/blitz/admin/api/v3/app/{appId}`.

Required permissions: `blitz_api_sys_app`.

As a result of executing the request, Blitz Identity Provider will return a JSON containing the application settings..

Request example:

```
GET /blitz/admin/api/v3/app/test-app HTTP/1.1
Authorization: Bearer cNw...Nz
```

Response example:

```
HTTP/2 200
...
content-type: application/json
etag: 96_1658847045000

{
  "name": "...",
  "tags": [
    "tag1",
    "tag2"
  ],
  "domain": "...",
  "startPageUrl": "...",
  "oauth": {
    "clientSecret": "...",
    "redirectUriPrefixes": [...],
    "predefinedRedirectUri": "...",
    "availableScopes": [..., "..."],
    "defaultScopes": [...],
    "enabled": true,
    "autoConsent": true,
    "idToken": {"claims": [...]},
    "accessTokenTtl": 3600,
    "defaultAccessType": "online",
    "refreshTokenTtl": 86400,
    "dynReg": {
      "isAllow": true,
      "allowedPlainJsonClaims": ["device_type"]
    },
    "pixyMandatory": true,
    "deviceGrant": {
      "userCodeFormat": "[0-9]{3,3}-[0-9]{3,3}-[0-9]{3,3}",
      "userCodeTtl": 120,
      "verificationUrl": "...",
      "useCompleteUri": true
    },
    "teAuthMethod": "client_secret_basic",
    "grantTypes": ["authorization_code", "client_credentials"],
    "responseTypes": ["code"],
    "extraClientSecret": "...",
```

(continues on next page)

(continued from previous page)

```

    "accessTokenFormat": "jwt",
    "logout": {
      "logoutAutoConsent": false,
      "logoutUriPrefixes": ["..."],
      "predefinedLogoutUri": "...",
      "frontchannelLogoutUri": "...",
      "frontchannelLogoutSessionRequired": true,
      "backchannelLogoutUri": "..."
    }
  },
  "simple": {
    "ssl": true,
    "formSelector": "...",
    "loginSelector": "...",
    "logoutUrl": "...",
    "postLogoutUrl": "..."
  },
  "rest": {
    "Basic": {"pswd": "..."},
    "TLS": []
  },
  "theme": "default",
  "saml": {
    "spMetadata": "...",
    "spAttributeFilterPolicy": {
      "id": "test-app",
      "attributeRules": [{"attr": "...", "isPermitted": true}]
    },
    "saml2SSOProfile": {
      "signAssertions": "always",
      "encryptAssertions": "always",
      "encryptNameIds": "always",
      "includeAttributeStatement": true
    }
  }
}

```

The content of the response may differ depending on the settings set for the application and the configured connection protocols. The `saml`, `oauth`, `simple`, and `rest` blocks may be missing if the appropriate protocols for the application are not configured.

The service's response contains the `etag` header. The value from this header should be used in the `If-Match` header if you plan to call the application registration services, edit application settings, or delete the application after receiving the application settings. Using `etag` Blitz Identity Provider checks that no other changes were made to the configuration file on the server in parallel sessions (optimistic blocking) between the last receipt of `etag` and calling the settings change operation with `If-Match`.

When using SAML, the `spMetadata` setting will contain a Base64URL encoded metadata file for the application (Service Provider Metadata).

The names of the settings returned by the service correspond to the names in the configuration file `blitz.conf`.

If the application settings for the transmitted `appId` are not found, the Blitz Identity Provider server returns the error HTTP 404 Not found.

## Application registration

To register an application, you need to make a PUT request at `https://login.company.com/blitz/admin/api/v3/app/{appId}`.

Required permissions: `blitz_api_sys_app_chg`.

The `If-Match` header can be (optionally) added to the request, containing the last `etag` value received from the server.

The request body must contain the settings values of the registered application.

Request example:

```
PUT /blitz/admin/api/v3/app/test-app2 HTTP/1.1
Authorization: Bearer cNw...Nz
Content-Type: application/json
If-Match: 98_1658857264000

{
  "name": "...",
  "tags": [
    "tag1",
    "tag2"
  ],
  "domain": "...",
  "startPageUrl": "...",
  "oauth": {
    "clientSecret": "...",
    "redirectUriPrefixes": [...],
    "predefinedRedirectUri": "...",
    "availableScopes": [..., "..."],
    "defaultScopes": [...],
    "enabled": true,
    "autoConsent": true,
    "idToken": {"claims": [...]},
    "accessTokenTtl": 3600,
    "defaultAccessType": "online",
    "refreshTokenTtl": 86400,
    "dynReg": {
      "isAllow": true,
      "allowedPlainJsonClaims": ["device_type"]
    },
    "pixyMandatory": true,
    "deviceGrant": {
      "userCodeFormat": "[0-9]{3,3}-[0-9]{3,3}-[0-9]{3,3}",
      "userCodeTtl": 120,
      "verificationUrl": "...",
      "useCompleteUri": true
    },
    "teAuthMethod": "client_secret_basic",
    "grantTypes": ["authorization_code", "client_credentials"],
    "responseTypes": ["code"],
    "extraClientSecret": "...",
    "accessTokenFormat": "jwt",
    "logout": {
      "logoutAutoConsent": false,
      "logoutUriPrefixes": [...],
      "predefinedLogoutUri": "...",
      "frontchannelLogoutUri": "...",
      "frontchannelLogoutSessionRequired": true,
      "backchannelLogoutUri": "..."
    }
  }
}
```

(continues on next page)

(continued from previous page)

```

    },
    "simple": {
      "ssl":true,
      "formSelector":"...",
      "loginSelector":"...",
      "logoutUrl":"...",
      "postLogoutUrl":"..."
    },
    "rest": {
      "Basic":{"pswd":"..."},
      "TLS":[]
    },
    "theme":"default",
    "saml": {
      "spMetadata":"...",
      "spAttributeFilterPolicy": {
        "id":"...",
        "attributeRules":[{"attr":"...", "isPermitted":true}]
      },
      "saml2SSOProfile": {
        "signAssertions":"always",
        "encryptAssertions":"always",
        "encryptNameIds":"always",
        "includeAttributeStatement":true
      }
    }
  }
}

```

When registering an application running on SAML, you need to consider the following features:

- the contents of the application metadata encoded in the Base64URL format must be passed to `spMetadata`.
- in the `id` setting in the `spAttributeFilterPolicy`, you must pass the same `id` that is passed in the URL as the `appId`.

If registration is successful, the server will return `HTTP 200`, the current application data and the current value `etag`.

Response example:

```

HTTP/2 200
...
content-type: application/json
etag: 99_1658857631000

{
  "id":"test-app2",
  "name":"...",
  ...
  "oauth": {
    ...
  },
  ...
}

```

If, during application registration, it is found that the data in the configuration file on the server was changed between receiving the `etag` and calling registration, the server will return a response with the code `HTTP 412 Precondition Failed` and the error body:

```
{
  "type": "process_error",
  "error": "cas_mismatch",
  "desc": "cas_mismatch"
}
```

If an error occurred during application registration, the server will return a response with the code HTTP 400 Bad Request with a description of the error.

Example of a response with a registration error:

```
{
  "type": "input_error",
  "error": "wrong_values",
  "errors": [
    {
      "type": "input_error",
      "error": "json.error.mandatory.field",
      "desc": "json.error.expected.array",
      "pos": "oauth.redirectUriPrefixes"
    },
    ...
  ]
}
```

### Changing application settings

To change the application settings, you need to make a POST request to `https://login.company.com/blitz/admin/api/v3/app/{appId}`.

Required permissions: `blitz_api_sys_app_chg`.

The `If-Match` header should be added to the request, containing the last `etag` value received from the server.

The request body must contain the new values of the application settings you want to change. You must send the entire branch with the parameter to be changed. For example, if the parameter is at level #3, its parent parameters at levels #1 and #2 must also be sent. To delete a parameter, the entire branch with the `null` value for that parameter must be sent.

Example of changing an application tag:

```
POST /blitz/admin/api/v3/app/test-app HTTP/1.1
Authorization: Bearer cNw..Nz
Content-Type: application/json
If-Match: 98_1658857264000

{
  "tags": [
    "default",
    "2F"
  ]
}
```

If the change is successful, the server will return HTTP 200, the current values of the application settings and a new `etag`.

Response example:

```
HTTP/2 200
...
```

(continues on next page)



(continued from previous page)

```

content-type: application/json
etag: 99_1658857631000

{
  "name": "",
  "tags": [
    "default",
    "2F"
  ],
  "domain": "test.app1.ru",
  "id": "app1",
  "simple": {
    "formSelector": "select",
    "postLogoutUrl": "http://localhost",
    "ssl": true,
    "loginSelector": "select",
    "js": "dyMw==",
    "logoutUrl": "https://localhost"
  },
  "disabled": false
}

```

Example of deleting an application tags:

```

POST /blitz/admin/api/v3/app/test-app HTTP/1.1
Authorization: Bearer cNw...Nz
Content-Type: application/json
If-Match: 98_1658857264000

{
  "tags": null
}

```

Response example:

```

HTTP/2 200
...
content-type: application/json
etag: 99_1658857631000

{
  "name": "",
  "domain": "test.app1.ru",
  "id": "app1",
  "simple": {
    "formSelector": "select",
    "postLogoutUrl": "http://localhost",
    "ssl": true,
    "loginSelector": "select",
    "js": "dyMw==",
    "logoutUrl": "https://localhost"
  },
  "disabled": false
}

```

If, when editing the application, it is found that the data in the configuration file on the server was changed between receiving the etag and calling the edit, the server will return a response with the code HTTP 412 Precondition Failed and the error body:

```
{
  "type": "process_error",
  "error": "cas_mismatch",
  "desc": "cas_mismatch"
}
```

If an error occurred while editing the application that incorrect data was transmitted, the server will return a response with the code HTTP 400 Bad Request with a description of the errors.

Example of an error response:

```
{
  "type": "input_error",
  "error": "wrong_values",
  "errors": [
    {
      "type": "input_error",
      "error": "json.error.mandatory.field",
      "desc": "json.error.expected.array",
      "pos": "oauth.redirectUriPrefixes"
    },
    ...
  ]
}
```

### Deleting an application

To delete an application, you must make a request using the DELETE method at `https://login.company.com/blitz/admin/api/v3/app/{appId}`.

Required permissions: `blitz_api_sys_app_chg`.

The `If-Match` header should be added to the request, containing the last `etag` value received from the server.

Request example:

```
DELETE /blitz/admin/api/v3/app/test-app HTTP/1.1
Authorization: Bearer cNw..Nz
If-Match: 99_1658857631000
```

If the application is successfully deleted, the server returns HTTP 204.

If, when deleting the application, it is found that the data in the configuration file on the server was changed between receiving the `etag` and calling the deletion, the server will return a response with the code HTTP 412 Precondition Failed and the error body:

```
{
  "type": "process_error",
  "error": "cas_mismatch",
  "desc": "cas_mismatch"
}
```

### 3.5.4 Invoking a third-party user registration application

In Blitz Identity Provider, you can configure the use of a third-party user registration application. In this case, Blitz Identity Provider will be able to call the user registration application from the login page (when clicking on the link *Register*) or as a result of the user's first login through an external identification provider. At the same time, the following features are available:

- If registration is started as a result of the first login through an external identification provider, then Blitz Identity Provider will transfer the attributes received from the external identification provider to the registration application. The application will be able to use them to pre-fill out the registration form.
- If the user successfully completes the registration, he will be able to continue the login process. For example, you can provide an automatic login of a registered user to the application in the same way as it happens when using the registration application built into Blitz Identity Provider.

To connect to Blitz Identity Provider a third-party registration application, it is necessary to support the services on the side of the registration web application in accordance with the requirements described in the following sections.

#### Registration Initiation Service

A third-party registration application must provide an HTTP POST service to initiate registration.

**Note:** The address of the service is set in the Blitz Identity Provider settings (see [Administration](#) (page 9)).

The service must accept the following parameters (in the form of JSON):

- `id` – the ID of the registration application;
- `entryPoint` – information about the login point. The following values are possible:
  - `SOCIAL` – registration is triggered due to the entry of a new user through an external identification provider;
  - `WEB` – the user initiated the registration on his own (selected “Register” on the login page).
- `appId` is the identifier of the application that the user originally wanted to log in to, as a result of which the registration process started;
- `expires` – the expiration time of the registration application. Specified in Unix time, in seconds;
- `source` – the source of information about the user (in the case of obtaining information from an external login provider). Contains the ID of the external login provider;
- a list of attributes obtained from an external identification provider. Attributes are passed from the account binding settings of the corresponding external identity provider.
- `hints` – hints passed to the login form call. For example, the user's login can be passed here, if the user initiated self-registration from the login form, which in turn was opened with the `login_hint` parameter;
- `lang` – the current language of the user interface on the login page.

Request example (when the user clicks “Register” on the login page):

```
POST /reg/url HTTP/1.1
Content-Type: application/json

{
  "id": "6DXDHyyiZ2hByUN-sCRUEdvAoQun7WwQ",
  "entryPoint": "WEB",
  "appId": "portal",
  "expires": 1608129702,
  "hints": {},
```

(continues on next page)

(continued from previous page)

```
"attrs": {},
"lang": "ru"
}
```

In response, the registration initiation service must return either an HTTP response to be executed in the user's browser (for example, the HTML code of the page or initiate redirection of the user in the browser to the registration page), or an error message.

Response example:

```
HTTP/1.1 302 Found
Location: https://www.company.ru/register/
```

As a result, the user will be redirected from Blitz Identity Provider to a third-party registration application.

### Registration completion service

When the user in the third-party registration application has entered all the data necessary for account registration, the third-party registration application should call the Blitz Identity Provider service to complete the registration of the user account. The service is called by the POST method at `https://login.company.com/blitz/reg/api/v1/users/{id}`, where the ID of the registration request previously received from Blitz Identity Provider is passed as the `id` in the URL of the service.

The following header should be added to the request, where `secret` is assigned to the application when registering in Blitz Identity Provider `client_id:rest_secret` in Base64 format:

```
Authorization: Basic <secret>
```

**Attention:** The list of attributes is provided as a sample. The contents of the list must be adjusted depending on the specific settings made during the implementation of Blitz Identity Provider. See [Administration](#) (page 9).

The request body must contain the attributes of the account being registered:

- `first_name` is a surname;
- `name` is the name;
- `middle_name` is a middle name;
- `phone_number` is a mobile phone number in the form of a composite object with attributes:
  - `value` is a phone number in the format `(country code)XXXXXXXXXX`;
  - `verified` – indicates that the phone has been verified – `true` or `false`;
- `email` – an email address in the form of a composite object with attributes:
  - `value` – email address;
  - `verified` – indicates that the address has been verified – `true` or `false`;
- `password` is the password for the user account being created (must match the configured password policy).

Request example (registration with confirmed email and phone number):

```
POST /blitz/reg/api/v1/users/6DXDHyyiZ2hByUN-sCRUEdvAoQun7WwQ HTTP/1.1
Authorization: Basic YXBwX2lkOmFwcF9zZWNYZXQ=
Content-Type: application/json
```

(continues on next page)

(continued from previous page)

```
{
  "first_name": "Иванов",
  "name": "Иван",
  "middle_name": "Иванович",
  "phone_number": {
    "value": "79991234567",
    "verified": true
  },
  "email": {
    "value": "mail@example.com",
    "verified": true
  },
  "password": "QWErty$123"
}
```

In response, Blitz Identity Provider, if registration is successful, will return JSON with the following data:

- `subject` is the ID of the registered user;
- `origin` is the link to which the user's browser should be directed;
- `cookies` are cookies that need to be set when redirecting the user's browser to a shared c Blitz Identity Provider domain;
- `instanceId`, `instructions` – other process information that should be ignored.

Response example:

```
{
  "instanceId": "amRiY2lkG9zdGdyZXM6YzhjMGEeYzEtYzdmYS00ZDg3LWFiYmMtZTNiYzg1YTk4",
  "subject": "5cffd68f-2cb8-4f7a-b0f3-9fa69a1fbbcd",
  "context": "6DXDHyyiz2hByUN-sCRUEdvAoQun7WwQ",
  "cookies": [{
    "name": "css",
    "value": "TSQA-AruOjUNphGZ984eLgzT_ROebNiBsyyjEg4n-nL-PdsiXqq"
  }],
  "origin": "/blitz/profile?",
  "instructions": []
}
```

After redirecting the user's browser registration by a third-party application using the link specified in `origin` and with the specified Blitz Identity Provider `cookies` will create a session and ensure that the user logs into the application for which the user has registered an account.

### 3.5.5 Authentication API

As a standard, if necessary, to identify and authenticate the user, the website or mobile application interacts with Blitz Identity Provider using any of the available protocols (see [Selecting an interaction protocol](#) (page 294)). At the same time, the application does not directly authenticate. The application redirects the user to Blitz Identity Provider to the login page. Next, Blitz Identity Provider independently offers the user various authentication methods, interacts with the user during the login process.

In some cases, it may be desirable to provide the user with the opportunity to complete identification and authentication without being redirected to the Blitz Identity Provider login page. Such capabilities are limited (not all login and login confirmation methods are available without redirection), and require a large amount of improvements on the application side (since the application needs to support the processing of various authentication-related scenarios).

Blitz Identity Provider provides an HTTP API that allows you to embed user identification and authentication into the application's web page without redirecting the user to a separate login page. This HTTP API is designed for

web applications. When using the API, a Web Single Sign-On is provided, namely, when the user subsequently logs in to another application connected to Blitz Identity Provider in the same web session, he will not be asked to log in again.

### Settings for using the API

The application must be registered in Blitz Identity Provider. The application in Blitz Identity Provider must be assigned `client_id` and `client_secret`, and the application return URL must be registered in Blitz Identity Provider.

The interaction of the application page and Blitz Identity Provider is based on the execution of a series of AJAX interactions. To enable such interaction, the following CORS (Cross-origin resource sharing) settings must be made on the application's web server and on the Blitz Identity Provider web server:

1. On the Blitz Identity Provider server, for the `/blitz/oauth/ae` handler, you need to configure the CORS permission by adding the following HTTP Headers (you need to specify the `origin` for the PROD site and the necessary `origin` for the required test environments):

```
"Access-Control-Allow-Origin" -> "https://{app-domain}",  
"Access-Control-Allow-Credentials" -> "true"
```

In this header, `{app-domain}` is the application domain.

2. On the portal server, the following CORS permission must be configured for the callback handler (see [Interaction scheme](#) (page 433)) of the response from Blitz Identity Provider (the permission is `null`, since after the redirect the browser resets `origin`):

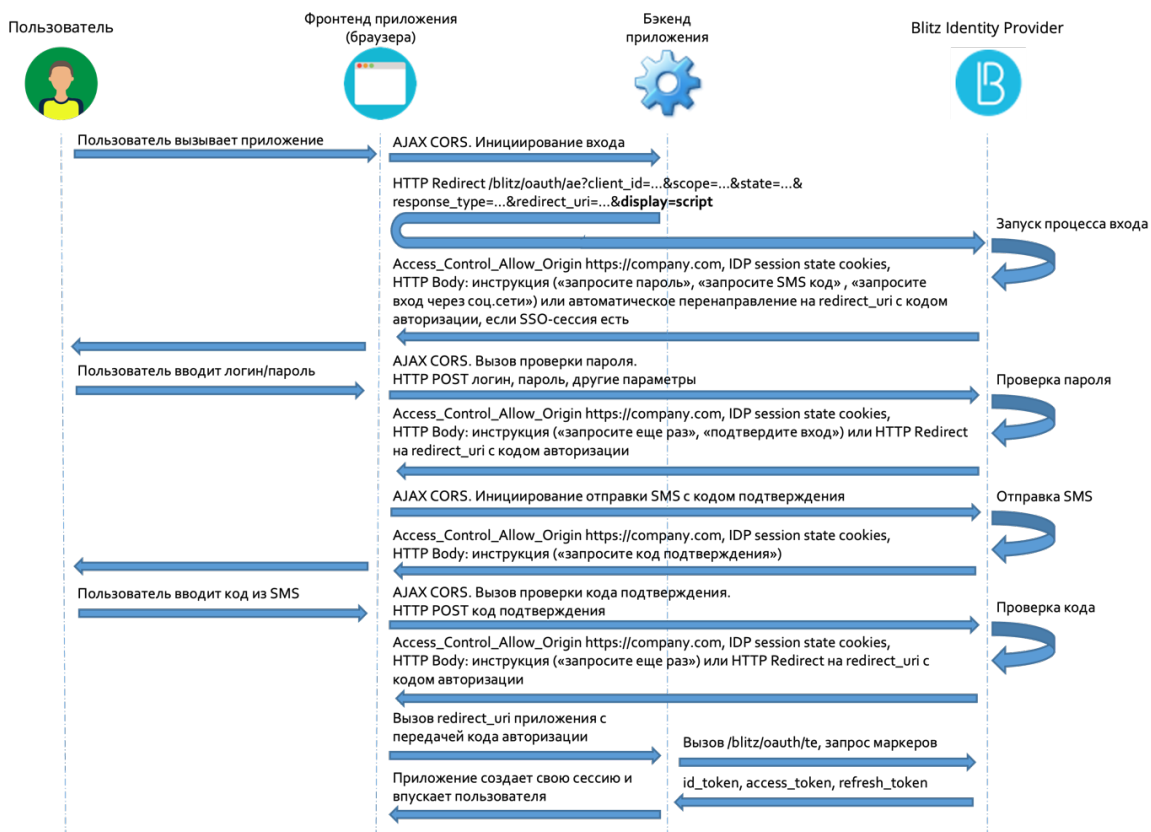
```
"Access-Control-Allow-Origin" -> null,  
"Access-Control-Allow-Credentials" -> "true"
```

### Interaction scheme

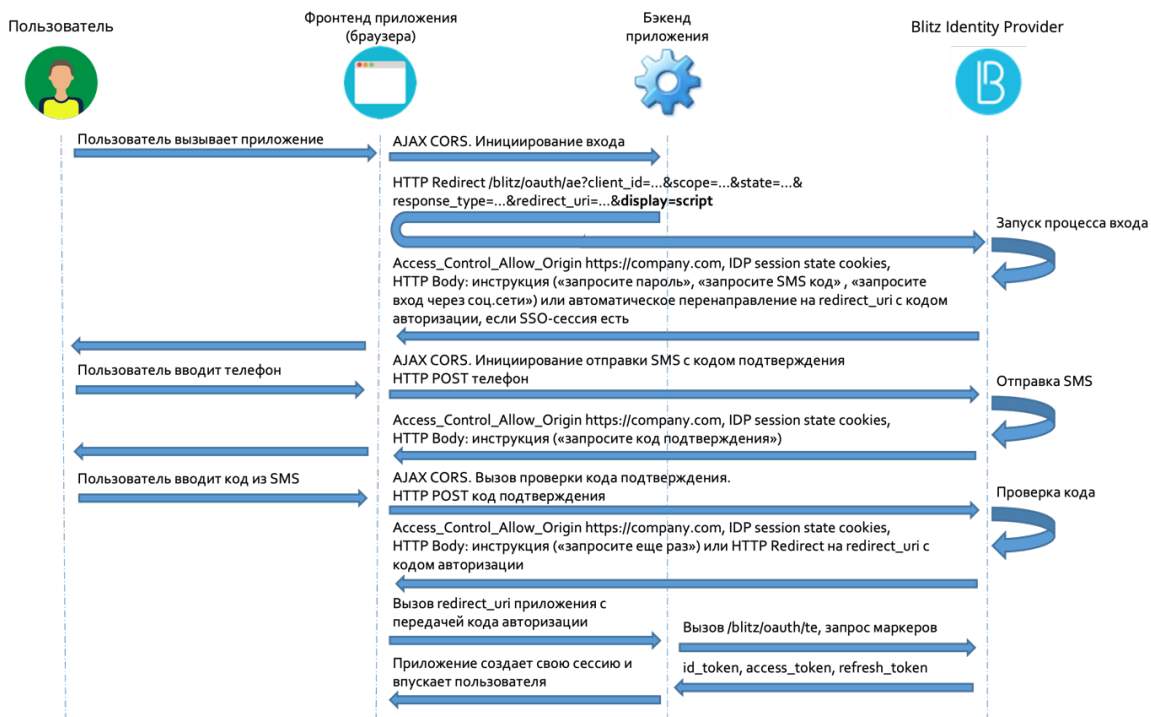
The HTTP authentication API allows you to:

- Check for an SSO session. If there is no SSO session, get a list of authentication methods available to the user.
- Perform identification and authentication using a username and password.
- Perform identification and authentication using a login (phone) and a confirmation code sent by SMS.
- Perform identification and authentication using a QR code;
- Confirm the login using the confirmation code sent by SMS.

The figure below shows an interaction scheme when logging in with a username and password, followed by confirmation of login using a confirmation code sent via SMS.



The following figure shows the interaction scheme when logging in by phone and the confirmation code sent by SMS.



The web application interacts with Blitz Identity Provider by executing a series of AJAX requests.

**Note:** Requests must be made with the saving and transfer of cookies – you must use `withCredentials: true`

The following sections describe the requests being called, possible responses, and recommendations for processing them. Examples of requests and responses are provided in the form of cURL calls.

### Starting the login process

To start the login process, the application must send an HTTP GET request to AJAX to Blitz Identity Provider (necessarily with `withCredentials: true`) to the usual Authorization Endpoint handler (`/blitz/oauth/ae`, see [Getting the authorization code](#) (page 300)), adding the special parameter `display=script.` to the request

Request example:

```
curl -v -b cookies.txt -c cookies.txt \
--request GET 'https://login.company.com/blitz/oauth/ae?response_type=code&client_
↪id=ais&scope=openid&state=...&display=script&redirect_uri=https%3A%2F%2Fapp.
↪company.com%2Ffre'
```

If an SSO session already exists, Blitz Identity Provider will automatically redirect the user to the `redirect_uri` handler address, adding the authorization code and the `state` parameter to the request. Using the received authorization code, the application will continue the standard OpenID Connect interaction to receive security tokens and account data.

Example of a redirect response if the SSO session already exists:

```
...
< HTTP/2 302
...
< location: https://...?code=...&state=...
...
```

An example of a response if authentication is required:

```
{
  "inquire": "choose_one",
  "items": [
    {
      "inquire": "login_with_password"
    },
    {
      "inquire": "request_auth_with_fed_point",
      "fp": "esia:esia_1"
    },
    ...
    {
      "inquire": "request_auth_with_fed_point",
      "fp": "yandex:yandex_1"
    },
    {
      "inquire": "login_to_send_sms"
    },
    {
      "inquire": "show_qr_code",
      "link": "https://...?code=dde087f0-8f4a-478e-886b-5354b0283362",
      "expires": 1660905165,
      "logo": "https://..."
    }
  ]
}
```

(continues on next page)



(continued from previous page)

```
]
}
```

If authentication is required, Blitz Identity Provider returns one of the possible instructions to the application:

- `login_with_password` – log in with your username and password;
- `request_auth_with_fed_point` – log in using an external identification provider (social network);
- `login_to_send_sms` – log in using your username and confirmation code sent via SMS;
- `show_qr_code` – display a QR code that allows you to log in.

If any of the authentication methods are not configured in Blitz Identity Provider or are unavailable for logging into the requesting application (for example, as a result of the settings of the “login procedure” for the corresponding application), then instructions on them will be missing in the service response.

Depending on the security modes included in Blitz Identity Provider, the `login_with_password` instruction may contain additional parameters:

- If the CAPTCHA mode is configured in Blitz Identity Provider when logging in, then the instructions will contain the `captchaId` parameter that the application needs to use for the CAPTCHA test:

```
{
  "inquire": "choose_one",
  "items": [
    {
      "inquire": "login_with_password",
      "captchaId": "9cf48a75-6be1-4008-b34e-8906220c472f"
    },
    ...
  ]
}
```

- If the password protection mode is configured in Blitz Identity Provider, which requires the application to solve a long-term computational task (Proof of Work), then the `proofOfWork` parameter will be in the instructions:

```
{
  "inquire": "choose_one",
  "items": [
    {
      "inquire": "login_with_password",
      "captchaId": "9cf48a75-6be1-4008-b34e-8906220c472f",
      "proofOfWork": "1:15:220313184752:abe...539::Ekf...w=="
    }
  ]
}
```

- If you receive the `proofOfWork` parameter, it is recommended to asynchronously immediately run the algorithm for finding a solution, without waiting for the user to select the login and password login mode and enter the data. This will hide the delay time for solving the problem from the user (it can be several seconds, depending on the complexity of the task). The [Hashcash<sup>89</sup>](http://www.hashcash.org) algorithm is currently being used.

**Important:** You need to supplement the `proofOfWork` parameter with such a value that the hash calculated from it using the SHA-1 algorithm contains at the beginning as many zero bits as specified by the task condition (the number after the first character `:` in the `proofOfWork` parameter).

<sup>89</sup> <http://www.hashcash.org>

For example, the solution for `1:15:yyyy03Su212003:BlitzIdp::McMybZIhxKXu57jd:0` will be the line `1:15:yyyy03Su212003:BlitzIdp::McMybZIhxKXu57jd:3/g`

Depending on the authentication method selected, the application calls in Blitz Identity Provider login using one of the following methods:

- [Login with the username and password](#) (page 437).
- [Login by phone and SMS confirmation code](#) (page 442).
- [Login using the QR code](#) (page 446).
- Login via an external identity provider – this type of login is possible only through a browser with redirection of the user to the login page of the external identity provider. You need to repeat the `Authorization Endpoint` call (see [Getting the authorization code](#) (page 300)), use the required value of the `bip_action_hint` parameter in the call, corresponding to the external login provider selected by the user (for example, `bip_action_hint=externalIdps:esia:esia_1`).

Request example:

```
https://login.company.com/blitz/oauth/ae?response_type=code&client_id=portal.ru&
↪scope=openid+profile&redirect_uri=https://apitest.company.com/success&
↪state=342a2c0c-d9ef-4cd6-b328-b67d9baf6a7f& bip_action_hint=used_
↪externalIdps:esia:esia_1
```

In this case, the completion of the login process will occur in a standard way in accordance with OpenID Connect – Blitz Identity Provider will return the authorization code to the `redirect_uri` application handler.

### Logging in using login and password

If CAPTCHA is configured in Blitz Identity Provider, then before calling the login and password verification, the application must make calls to receive and verify the CAPTCHA. Verification requests should be generated through specialized proxy services Blitz Identity Provider, and not directly to CAPTCHA services.

When using reCAPTCHA v3, you must initialize reCAPTCHA v3 according to the [documentation](#)<sup>90</sup>.

- Upload the script on the application page using the same reCAPTCHA v3 `sitekey` as registered in Blitz Identity Provider:

```
<script src="https://www.google.com/recaptcha/api.js?render=reCAPTCHA_site_key"></
↪script>
```

- Call `grecaptcha.execute` on pressing the login button:

```
<script>
  function onClick(e) {
    e.preventDefault();
    grecaptcha.ready(function() {
      grecaptcha.execute('reCAPTCHA_site_key', {action: 'submit'}).
↪then(function(token) {
      // Add your logic to submit to your backend server here.
    });
  });
}
</script>
```

Immediately after calling from the reCAPTCHA services login page, you must call the `verify` operation from the application server. The call should not be made directly to the Google servers, but through a special proxy service in Blitz Identity Provider.

<sup>90</sup> [https://developers.google.com/recaptcha/docs/v3#programmatically\\_invoke\\_the\\_challenge](https://developers.google.com/recaptcha/docs/v3#programmatically_invoke_the_challenge)

Example of a verification request (operation `verify`):

```
POST /blitz/login/captcha/verify
Content-Type: 'text/json'
{
  "ctx": {
    // captchaId
    "id": "9cf48a75-6be1-4008-b34e-8906220c472f",
    "method": "password"
  },
  "params": {
    // token для проверки капчи, полученный при регистрации в Google
    "response": "03...sA"
  }
}
```

Ответ ``HTTP 200 OK``:

```
{
  "action": "submit",
  "challenge_ts": "2021-03-16T11:18:41Z",
  "success": true,
  "hostname": "company.com",
  "score": 0.9
}
```

Besides, if Proof of Work protection is enabled in Blitz Identity Provider, then you need to calculate the value of the `proofOfWork` parameter (see [Starting the login process](#) (page 435)).

To verify the login and password, the application must send an HTTP POST request to AJAX to Blitz Identity Provider (necessarily with `withCredentials: true`) on the URL `https://login.company.com/blitz/login/methods/headless/password` with `Content-Type x-www-form-urlencoded` and a Body containing the `login` and `password` parameters, as well as the calculated `proofOfWork` (if this parameter was received from Blitz Identity Provider when starting the login process).

Request example:

```
curl -v -b cookies.txt -c cookies.txt \
--request POST 'https://login.company.com/blitz/login/methods/headless/password' \
--header "Content-Type: application/x-www-form-urlencoded" \
--data 'login=логин&password=пароль&proofOfWork=решение'
```

Blitz Identity Provider performs the necessary security checks upon receipt of the request (whether the CAPTCHA has been passed, whether `ProofOfWork` has been resolved, whether the account has been blocked). If the security checks are passed, then Blitz Identity Provider checks the transmitted username and password.

If the login and password checks are successful and if the authentication is sufficient, Blitz Identity Provider will automatically redirect the user to the `redirect_uri` handler address, adding the authorization code and the `state` parameter to the request. Using the received authorization code, the application will continue the standard OpenID Connect interaction to receive security tokens and account data.

Example of a redirect response if the SSO session already exists:

```
...
< HTTP/2 302
...
< location: https://...?code=...&state=...
...
```

If any checks failed or if further actions from the user are required, then Blitz Identity Provider returns one of the instructions.

Example of a response in case of a login and password verification error:

```
{
  "inquire": "login_with_password",
  "errors": [
    {
      "code": "invalid_credentials",
      "params": {}
    }
  ]
}
```

Upon receiving such a response, the application can display the error text and prompt the user to enter another username and password, after which you can repeat the login and password verification.

If the user has entered a password that was previously in the account, or if the account is blocked, the error will look like:

```
{
  "inquire": "login_with_password",
  "captchaId": "9cf48a75-6be1-4008-b34e-8906220c472f",
  "proofOfWork": "1:15:220313184752:abe...539::Ekf...w==:",
  "errors": [
    {
      "code": "invalid_credentials",
      "params": {
        "_cause": "used_old_password"
      }
    }
  ]
}
```

An example of getting an error that the CAPTCHA check failed:

```
{
  "inquire": "login_with_password",
  "captchaId": "9cf48a75-6be1-4008-b34e-8906220c472f",
  "errors": [
    {
      "code": "invalid_captcha",
      "params": {}
    }
  ]
}
```

An example of an error that the Proof of Work solution was not checked:

```
{
  "inquire": "handle_error",
  "errors": [
    {
      "code": "doesNotMatch",
      "params": {}
    }
  ]
}
```

If special protection is enabled in Blitz Identity Provider to delay the verification of the login and password, then when checking the login and password, you can receive the following instruction from Blitz Identity Provider that you need to re-call the password verification after a certain number of seconds:

```
{
  "inquire": "delayed_login_with_password",
  "delayedFor": 5
}
```

A repeat call must be made when the required time has passed. The `isDelayed=true` parameter must be passed to the repeated call.

Example of calling password verification again in response to the instruction `delayed_login_with_password`:

```
curl -v -b cookies.txt -c cookies.txt \
--request POST 'https://login.company.com/blitz/login/methods/headless/password' \
--header "Content-Type: application/x-www-form-urlencoded" \
--data 'login=логин&password=пароль&proofOfWork=решение&isDelayed=true'
```

If special password brute force protection is enabled in Blitz Identity Provider, then Blitz Identity Provider may request additional CAPTCHA verification when verifying the password for this account. There are two possible situations:

- The user passed the wrong password, after which the protection turned on, and the CAPTCHA is needed for another authentication attempt.
- User account password brute force protection was enabled earlier. The current transmitted password was not verified because the CAPTCHA test was not performed.

In the first case, you need to inform the user that the username and password are incorrect, and for a new attempt, in addition to entering the password, request to take a CAPTCHA test.

In the second case, you need to ask the user to take a CAPTCHA test, and then send the previously entered username and password for verification.

An example of the answer for the first case is that the password is incorrect and a CAPTCHA test is needed:

```
{
  "inquire": "login_with_password",
  "captchaId": "1c9e4047-c8c4-47ad-a447-cc1809bd3e6c",
  "errors": [
    {
      "code": "invalid_credentials",
      "params": {}
    }
  ]
}
```

An example of the answer for the second case is that the password was not checked and a CAPTCHA test is needed:

```
{
  "inquire": "login_with_password",
  "captchaId": "2f818f5d-3a89-428d-b424-cde38c19051e",
  "errors": [
    {
      "code": "bypass_captcha",
      "params": {}
    }
  ]
}
```

An example of an error if the account is temporarily blocked:

```
{
  "inquire": "login_with_password",
  "errors": [
    {
      "code": "pswd_method_temp_locked",
      "params": {"0": "2"}
    }
  ]
}
```

An example of an error if the account is blocked due to prolonged inactivity:

```
{
  "inquire": "handle_error",
  "errors": [
    {
      "code": "inactivity_lock",
      "params": {}
    }
  ]
}
```

If the account password does not match the password policy, it may be necessary to change the password when logging in. In this case, Blitz Identity Provider will return instructions that you need to redirect the user to the page with the specified address:

```
{
  "inquire": "go_to_web",
  "redirect_uri":
    "https://.../blitz/login/methods/password/change?f=false&c=password_policy_
    ↪violated"
}
```

If the login and password are successful, but you additionally need to confirm the login, then instructions will be returned with possible confirmation methods:

```
{
  "inquire": "choose_one",
  "items": [
    {
      "inquire": "ask_to_send_sms"
    },
    {
      "inquire": "go_to_web",
      "redirect_uri": "https://login.company.com/blitz/login/methods2/sms"
    }
  ]
}
```

You can either redirect the user to a web page so that he continues to confirm login on the Blitz Identity Provider web page, or continue to use the HTTP API to [confirm login by SMS code](#) (page 448).

If the login procedure set for the application is configured to invoke an additional screen after logging in (for example, see [Invoking the auxiliary application at the moment of login](#) (page 419)). Invoking the auxiliary application at the time of login, then Blitz Identity Provider redirects the user to this screen.

### Login by phone and confirmation code

Logging in by phone and confirmation code consists of the following steps:

- Sending a confirmation code to the user via SMS.
- Verification of the confirmation code entered by the user.

To send the user a confirmation code via SMS, the application must send an HTTP POST request to AJAX to Blitz Identity Provider (necessarily with `withCredentials: true`) on the URL `https://login.company.com/blitz/login/methods/headless/sms/bind` with `Content-Type x-www-form-urlencoded` and a `Body` containing the user's login. It is recommended to pass the phone number entered by the user as the `login`.

Request example:

```
curl -v -b cookies.txt -c cookies.txt \
--request POST 'https://login.company.com/blitz/login/methods/headless/sms/bind' \
--header "Content-Type: application/x-www-form-urlencoded" \
--data 'login=логин'
```

If the account with the transmitted username is not found, then Blitz Identity Provider returns an error:

```
{
  "inquire": "handle_error",
  "errors": [
    {
      "code": "no_subject_found",
      "params": {}
    }
  ]
}
```

If the account is found, but a search of confirmation codes was previously recorded for it, then Blitz Identity Provider returns an error:

```
{
  "inquire": "handle_error",
  "errors": [
    {
      "code": "method_temp_locked",
      "params": {}
    }
  ]
}
```

If the account is found and it is possible to log in this way, then Blitz Identity Provider will send the user an SMS with a confirmation code and return a response:

```
{
  "inquire": "enter_sms_code",
  "contact": "+79991234567",
  "ttl": 300,
  "remain_attempts": 3
}
```

The received response indicates how many seconds the user has left to send the code for verification (`ttl`), how many attempts he has to enter the code (`remain_attempts`), to which phone number the code was sent to him (`contact`).

To verify the confirmation code entered by the user, the application must send an HTTP POST request to AJAX to Blitz Identity Provider (necessarily with `withCredentials: true`) on the URL `https://`

/login.company.com/blitz/login/methods/headless/sms/bind with Content-Type x-www-form-urlencoded and a Body containing an sms-code with a confirmation code.

Request example:

```
curl -v -b cookies.txt -c cookies.txt \
--request POST 'https://login.company.com/blitz/login/methods/headless/sms/bind' \
--header "Content-Type: application/x-www-form-urlencoded" \
--data 'sms-code=123456'
```

If the code is incorrect, then Blitz Identity Provider will return an error:

```
{
  "inquire": "handle_error",
  "errors": [
    {
      "code": "invalid_otp",
      "params": {}
    }
  ],
  "contact": "+79991234567",
  "remain_attempts": 2,
  "ttl": 276
}
```

If the number of code verification attempts has ended, Blitz Identity Provider returns an error:

```
{
  "inquire": "handle_error",
  "errors": [
    {
      "code": "no_attempts",
      "params": {}
    }
  ]
}
```

If the code has expired, Blitz Identity Provider returns an error:

```
{
  "inquire": "handle_error",
  "errors": [
    {
      "code": "expired",
      "params": {}
    }
  ]
}
```

In case of this error, you can request to send a new confirmation code. To do this, the application must call Blitz Identity Provider as follows:

```
curl -v -b cookies.txt -c cookies.txt \
--request POST 'https://login.company.com/blitz/login/methods/headless/sms/bind' \
--header "Content-Type: application/x-www-form-urlencoded" \
--data 'sms-send=sms'
```

If you request to resend the code before the expiration of the previous one, an error will be returned:



```
{
  "inquire": "handle_error",
  "errors": [
    {
      "code": "code_not_expired",
      "params": {}
    }
  ]
}
```

If the total number of attempts to log in using the confirmation code from the SMS is exceeded, then Blitz Identity Provider temporarily blocks the login for the account using the confirmation code. In this case, the next time you try to enter an incorrect confirmation code, Blitz Identity Provider may return an error:

```
{
  "inquire": "handle_error",
  "errors": [
    {
      "code": "method_temp_locked",
      "params": {}
    }
  ]
}
```

If the entered confirmation code is correct, and this is enough to complete the login, then Blitz Identity Provider will automatically redirect the user to the `redirect_uri` handler address, adding the authorization code and the `state` parameter to the request. Using the received authorization code, the application will continue the standard OpenID Connect interaction to receive security tokens and account data.

Example of a response in case of a successful login:

```
...
< HTTP/2 302
...
< location: https://...?code=...&state=...
...
```

If the verification of the confirmation code is successful, but you additionally need to confirm the login, an instruction will be returned with possible confirmation methods:

```
{
  "inquire": "choose_one",
  "items": [
    {
      "inquire": "go_to_web",
      "redirect_uri": "https://login.company.com/blitz/login/methods2/email"
    }
  ]
}
```

## Logging in with email

Logging in using email consists of the following steps:

- Emailing a confirmation code to a user .
- Verification of the confirmation code entered by the user.

To send the user a confirmation code via email, the application must send an HTTP POST request to AJAX to Blitz Identity Provider (necessarily with `withCredentials: true`) on the URL `https://login.company.com/blitz/login/methods/headless/email/bind` with `Content-Type x-www-form-urlencoded` and a `Body` containing the user's login. It is recommended to pass the email entered by the user as the `login`.

Request example:

```
curl --location --request POST 'https://login.company.com/blitz/login/methods/
↪headless/email/bind' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'login=<email>'
```

Response example:

```
{
  "inquire": "enter_email_code",
  "contact": "user@gmail.com",
  "remain_attempts": 3,
  "ttl": 300
}
```

Code verification:

```
curl --location --request POST 'https://login.company.com/blitz/login/methods/
↪headless/email/bind' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'email-code=746234'
```

Response options if the verification failed:

```
{
  "errors": [
    {
      "code": "invalid_otp",
      "params": {}
    }
  ],
  "contact": "user@gmail.com",
  "inquire": "handle_error",
  "remain_attempts": 2,
  "ttl": 257
}
```

```
{
  "inquire": "handle_error",
  "errors": [
    {
      "code": "no_attempts",
      "params": {}
    }
  ]
}
```

Resubmitting the code:

```
curl --location --request POST 'https://login.company.com/blitz/login/methods/
↪headless/email/bind' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--header 'Cookie: blc=Hc.; bua=7cd2c312-...; cTm=1:RGVm==; cTmTgs=1:c3Nv; oauth_
↪az=0MyeV-5v_...OnIE; portal_lang=ru' \
--data-urlencode 'email-send=email'
```

Response to the code resubmission:

```
{
  "inquire": "enter_email_code",
  "contact": "user@gmail.com",
  "remain_attempts": 1,
  "ttl": 288
}
```

### Login by QR code

Login by QR code consists of the following steps:

- Displaying a QR code to the user on the computer where the login is performed;
- Periodic check whether the user has scanned the QR code with the mobile application;
- Periodic verification of whether the user has confirmed or rejected the QR code login request in the mobile application;
- Updating an outdated QR code.

The application should display the QR code to the user by encoding the string received from Blitz Identity Provider into it. Below is a fragment of the instructions for logging in using a QR code (see [Starting the login process](#) (page 435)).

```
{
  "inquire": "choose_one",
  "items": [
    ...
    {
      "inquire": "show_qr_code",
      "link": "https://...?code=dde087f0-8f4a-478e-886b-5354b0283362",
      "expires": 1660905165,
      "logo": "https://..."
    }
  ]
}
```

Explanations of the parameters received from Blitz Identity Provider:

- `inquire` is an instruction with an available login option, in case of login using a QR code, the value is `show_qr_code`;
- `link` is a link that should be encoded in a QR code displayed to the user;
- `expires` is the time (in Unix Epoch) until which the QR code is valid. After the expiration date, it is recommended to display to the user that the QR code is expired;
- `logo` – if Blitz Identity Provider is configured to display a small logo in the center on top of the QR code, then the URL of the logo will be returned in the specified setting.

When the application displays the QR code to the user, it is necessary to wait for the user to read the QR code with a special mobile application. The integration of the mobile application for embedding the QR code login function is described in [Login to the application using a QR code](#) (page 325).

The web application can periodically check whether the QR code has been read by the mobile application. To do this, you need to execute an HTTP GET request in AJAX to Blitz Identity Provider (necessarily with `withCredentials: true`) on the URL `https://login.company.com/blitz/login/methods/headless/qrCode/pull`.

Request example:

```
curl -v -b cookies.txt -c cookies.txt \
--request GET 'https://login.company.com/blitz/login/methods/headless/qrCode/pull'
```

If the QR code has not been read yet, the response will be returned:

```
{
  "command": "showQRCode"
}
```

If the QR code is read, the response will be returned:

```
{
  "command": "askForConfirm"
}
```

In this case, you can update the user's web page and write on it that confirmation of login in the mobile application is expected.

If the QR code is expired, the response will be returned:

```
{
  "command": "needRefresh",
  "cause": "qr_code_expired"
}
```

If the user rejected the QR code login request in the mobile application, the response will be returned:

```
{
  "command": "needRefresh",
  "cause": "refused_login"
}
```

If the QR code is expired or the user declined to log in using the QR code, then you can ask the user to get a new QR code. To do this, run an HTTP POST request in AJAX to Blitz Identity Provider (required with `withCredentials: true`) on the URL `https://login.company.com/blitz/login/methods/headless/qrCode/refresh`.

Request example:

```
curl -v -b cookies.txt -c cookies.txt \
--request POST 'https://login.company.com/blitz/login/methods/headless/qrCode/refresh'
```

Response example:

```
{
  "link": "https://...?code=4ddf1667-d57f-4f86-b8f2-3ee53b367dfe",
  "expires": 1660922807,
}
```

(continues on next page)

(continued from previous page)

```
"logo": "https://..."
}
```

If the user has confirmed the QR code login request in the mobile application, then the service `https://login.company.com/blitz/login/methods/headless/qrCode/pull` will return the response:

```
{
  "command": "needComplete"
}
```

In response to this request, to complete the login, an HTTP POST request must be executed in AJAX to Blitz Identity Provider (necessarily with `withCredentials: true`) on the URL `https://login.company.com/blitz/login/methods/headless/qrCode/complete`.

Request example:

```
curl -v -b cookies.txt -c cookies.txt \
--request POST 'https://login.company.com/blitz/login/methods/headless/qrCode/
↵complete'
```

If the authentication is sufficient to complete the login, Blitz Identity Provider will automatically redirect the user to the `redirect_uri` handler address, adding the authorization code and the `state` parameter to the request. Using the received authorization code, the application will continue the standard OpenID Connect interaction to receive security tokens and account data.

Example of a response in case of a successful login:

```
...
< HTTP/2 302
...
< location: https://...?code=...&state=...
...
```

If you need to go through additional login confirmation, instructions will be returned with possible confirmation methods:

```
{
  "inquire": "choose_one",
  "items": [
    {
      "inquire": "go_to_web",
      "redirect_uri": "https://login.company.com/blitz/login/methods2/email"
    }
  ]
}
```

### Confirmation of login by confirmation code

Confirmation of the login using the SMS confirmation code consists of the following steps:

- Sending a confirmation code to the user via SMS.
- Verification of the confirmation code entered by the user.

To send the user a confirmation code via SMS, the application must send an HTTP POST request to AJAX to Blitz Identity Provider (necessarily with `withCredentials: true`) on the URL `https://login.company.com/blitz/login/methods/headless/sms/bind` with `Content-Type x-www-form-urlencoded` without Body:

Request example:

```
curl -v -b cookies.txt -c cookies.txt \
--request POST 'https://login.company.com/blitz/login/methods/headless/sms/bind' \
--header "Content-Type: application/x-www-form-urlencoded"
```

Blitz Identity Provider will send the user an SMS with a confirmation code and return a response:

```
{
  "inquire": "enter_sms_code",
  "contact": "+79991234567",
  "ttl": 300,
  "remain_attempts": 3
}
```

The received response indicates how many seconds the user has left to send the code for verification (ttl), how many attempts he has to enter the code (remain\_attempts), to which phone number the code was sent to him (contact).

To verify the confirmation code entered by the user, the application must send an HTTP POST request to AJAX to Blitz Identity Provider (necessarily with `withCredentials: true`) on the URL `https://login.company.com/blitz/login/methods/headless/sms/bind` with `Content-Type: application/x-www-form-urlencoded` and a `Body` containing an `sms-code` with a confirmation code.

Request example:

```
curl -v -b cookies.txt -c cookies.txt \
--request POST 'https://login.company.com/blitz/login/methods/headless/sms/bind' \
--header "Content-Type: application/x-www-form-urlencoded" \
--data 'sms-code=123456'
```

If the code is incorrect, then Blitz Identity Provider will return an error:

```
{
  "inquire": "handle_error",
  "errors": [
    {
      "code": "invalid_otp",
      "params": {}
    }
  ],
  "contact": "+79991234567",
  "remain_attempts": 2,
  "ttl": 276
}
```

If the number of code verification attempts has ended, Blitz Identity Provider returns an error:

```
{
  "inquire": "handle_error",
  "errors": [
    {
      "code": "no_attempts",
      "params": {}
    }
  ]
}
```

If the code has expired, Blitz Identity Provider returns an error:

```
{
  "inquire": "handle_error",
  "errors": [
    {
      "code": "expired",
      "params": {}
    }
  ]
}
```

In case of this error, you can request to send a new confirmation code. To do this, the application must call Blitz Identity Provider as follows:

```
curl -v -b cookies.txt -c cookies.txt \
--request POST 'https://login.company.com/blitz/login/methods/headless/sms/bind' \
--header "Content-Type: application/x-www-form-urlencoded" \
--data 'sms-send=sms'
```

If you request to resend the code before the expiration of the previous one, an error will be returned:

```
{
  "inquire": "handle_error",
  "errors": [
    {
      "code": "code_not_expired",
      "params": {}
    }
  ]
}
```

If the total number of attempts to confirm login by SMS confirmation code is exceeded, then Blitz Identity Provider temporarily blocks login confirmation for the account by confirmation code. In this case, the next time you try to enter an incorrect confirmation code, Blitz Identity Provider may return an error:

```
{
  "inquire": "handle_error",
  "errors": [
    {
      "code": "method_temp_locked",
      "params": {}
    }
  ]
}
```

If the entered confirmation code is correct, and this is enough to complete the login, then Blitz Identity Provider will automatically redirect the user to the `redirect_uri` handler address, adding the authorization code and the `state` parameter to the request. Using the received authorization code, the application will continue the standard OpenID Connect interaction to receive security tokens and account data.

Example of a response in case of a successful login:

```
...
< HTTP/2 302
...
< location: https://...?code=...&state=...
...
```

# Chapter 4

## Modules

In this section, you will find detailed information on the Blitz Identity Provider add-on modules.

### 4.1 Blitz Keeper security gateway

#### 4.1.1 About Blitz Keeper

With the Blitz Identity Provider, you can implement access control when secured services are invoked by applications.

Providing authorization when applications invoke services is based on OAuth 2.0 specifications. Before using services, an application must obtain an access token (`access_token`) from Blitz Identity Provider. [Various interaction methods](#) (page 294) are available to the application to obtain an access token. The access token can be obtained:

- in the context of a user login - the token will include information about the user and a set of scopes (permissions) granted by the user to the application;
- to the application outside of the user login - the token will include a set of scopes (permissions) from the granted to the application.

Then using the access token obtained, the application can invoke services. In doing so, the following complications will occur:

- within each service it will be necessary to implement its own authorization logic – check the provided access token, extract information about the user and provided consents (permissions) from them, and analyze whether those permissions are sufficient for service to be executed.
- the application will use a single access token to invoke different services. In this case, the access token may contain more information about the user and a larger set of consents (permissions) than is necessary for a particular invoked service. This will violate the principle of least privilege - the service will get more access rights than it needs to perform its task.

To solve the above described difficulties, Blitz Identity Provider provides a special application - the Security Gateway (`blitz-keeper`). This application is a specialized proxy server used when calling protected services - the application does not call the services directly, but through the Security Gateway. The Security Gateway takes care of the following tasks:

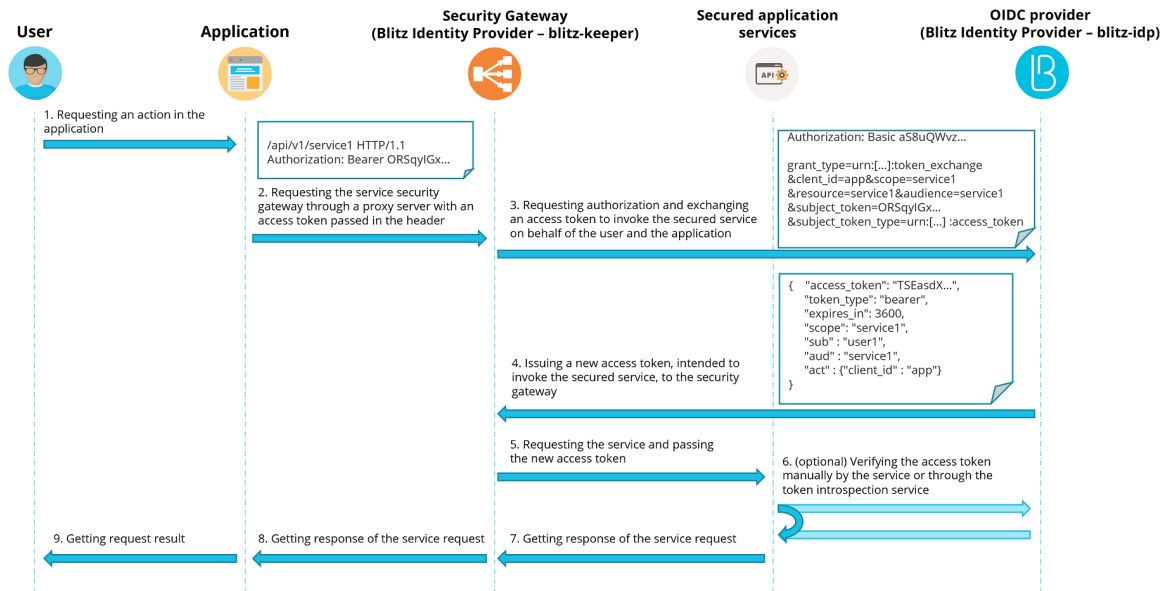
- Checks the authorization header included in the invoke of service, extracts the access token and, in interaction with the authorization service (`blitz-idp`), checks whether the access token is valid, and whether the user and the application has sufficient access rights to invoke the secured service.
- In interaction with the authorization service (`blitz-idp`) replaces the access token in such a way that the security token transmitted from the Security Gateway to the protected service contains only the set of user information and permissions required for the protected service operation. Redundant permission and



user information can be either removed from the security token or additional permissions and information added to the access token, if this is configured in the security policy.

- Logs successful and unsuccessful access control events in the Blitz Identity Provider security event log.

The interaction between the Security Gateway and the authorization service is based on the [OAuth 2.0 Token Exchange](#)<sup>91</sup> specification. Picture of the interaction is shown in the diagram.



Configuring the use of the security gateway to access protected services is described in the following sections.

#### 4.1.2 Installing the blitz-keeper service

**Important:** See the [system requirements](#) (page 10).

To install the `blitz-keeper` service, use the `blitz-keeper-5.X.X.bin` installer.

To install `blitz-keeper`, do the following:

1. Copy the `blitz-keeper-5.X.X.bin` file from the Blitz Identity Provider distribution to any directory on the designated server (for example, `/tmp`).
2. Run the `blitz-keeper-5.X.X.bin` installer:

```
cd /tmp
chmod +x blitz-keeper-5.X.X.bin
./blitz-keeper-5.X.X.bin
```

In response to the installer's questions, specify `JAVA_HOME` as the directory of the JDK installation on the server.

Blitz Keeper will be installed in the `/usr/share/identityblitz` directory.

3. Add the `blitz-keeper` service to autostart and launch it:

```
systemctl enable blitz-keeper
systemctl start blitz-keeper
```

4. Adjust the balancing settings block in the nginx configuration file (directory `/etc/nginx/conf.d`):

<sup>91</sup> <https://tools.ietf.org/html/rfc8693>

```

upstream blitz-keeper {
    server [BLITZ-KPR-NODE-01]:9012 max_fails=3 fail_timeout=120;
    server [BLITZ-KPR-NODE-02]:9012 max_fails=3 fail_timeout=120;
}

```

**Note:** [BLITZ-%%%N-NODE-XX] – names (hostname) of the blitz-keeper servers.

### 4.1.3 Configuring Blitz Keeper

Blitz Keeper is configured by editing the configuration file `blitz-keeper.conf` located in the `/etc/blitz-keeper` directory. Example of the configuration file:

```

{
  "authenticators": {
    "prod-auth": {
      "type": "token-exchange",
      "te": "https://blitz-host/blitz/oauth/te",
    },
  },
  "services" : {
    "api-1":{
      "display-name" : "secured services",
      "host": "service-host.com",
      "locations": {
        "/api/service1/**": {
          "methods" : ["GET", "POST"],
          "authenticator": "prod-auth",
          "required-scopes": ["scope1", "scope2"]
        },
        "/path/api/user/*/getdata/**": {
          "methods" : ["GET", "PUT"],
          "authenticator": "prod-auth",
          "required-scopes": ["scope3"]
        }
      }
    }
  }
}

```

In the `authenticators` block it is necessary to register all used `blitz-idp` authorization services. Usually it is sufficient to use one single authorization service to protect the services, and then only one block needs to be filled in as in the example (in the example one authorization service named `prod-auth` is registered). If several separate Blitz Identity Provider installations are used in the system (for example, PROD and TEST environment or internal loop for employees and external loop for clients), then you can use a common security gateway that will interact with several different authorization services - then you need to specify the settings of several authorization services in the `authenticators` block. For each authorization service a name is set (in the example `prod-auth` is used, but you can set any name). In the settings block of the authorization service the type of interaction (`type`) is set in the value `token-exchange` (so far it is the only supported type of interaction) and the address (`te`) of the call of the Token Endpoint handler of the authorization service. If `blitz-keeper` is deployed on separate servers, it is recommended to specify the address of the handler with `https` and domain name. If the `blitz-keeper` application is deployed on the same server as the `blitz-idp` authorization service, it is recommended to specify a local name in `te`, e.g. `http://localhost:9000/blitz/oauth/te`.

In the `services` block you must register the protected services. You can create a common settings block or several separate blocks for all protected services. Each block has a name (in the example, `api-1`). The settings are defined inside the block:

- `display-name` - text description of the service (any comment or description);

- `host` - server address of the secured service;
- `locations` - allowed paths and operations of service invoke.

The `locations` block specifies the settings of all service paths and allowed methods. The service address is specified as the name of each nested block. It is acceptable to use an asterisk (\*) in the address to indicate the omission of a separate component in the service path address and it is acceptable to use a double asterisk (\*\*) to indicate that the rest of the service path can be any component. Within the service address nested block, you can optionally list the allowed methods of the service (`methods` setting), specify the name of the authorization service to be used (`authenticator` setting) and a list of permissions (`required-scopes` setting) for the target access token to be included in the access token passed to the protected service.

After changing the settings in `blitz-keeper.conf`, the security gateway must be restarted.

#### 4.1.4 Creating service access rules

See the [general information](#) (page 282) on how to create the list of rules to access protected services over Token Exchange.

#### 4.1.5 Configuring access token exchange

See the [general information](#) (page 286) on how to configure access token exchange.

#### 4.1.6 Viewing logs

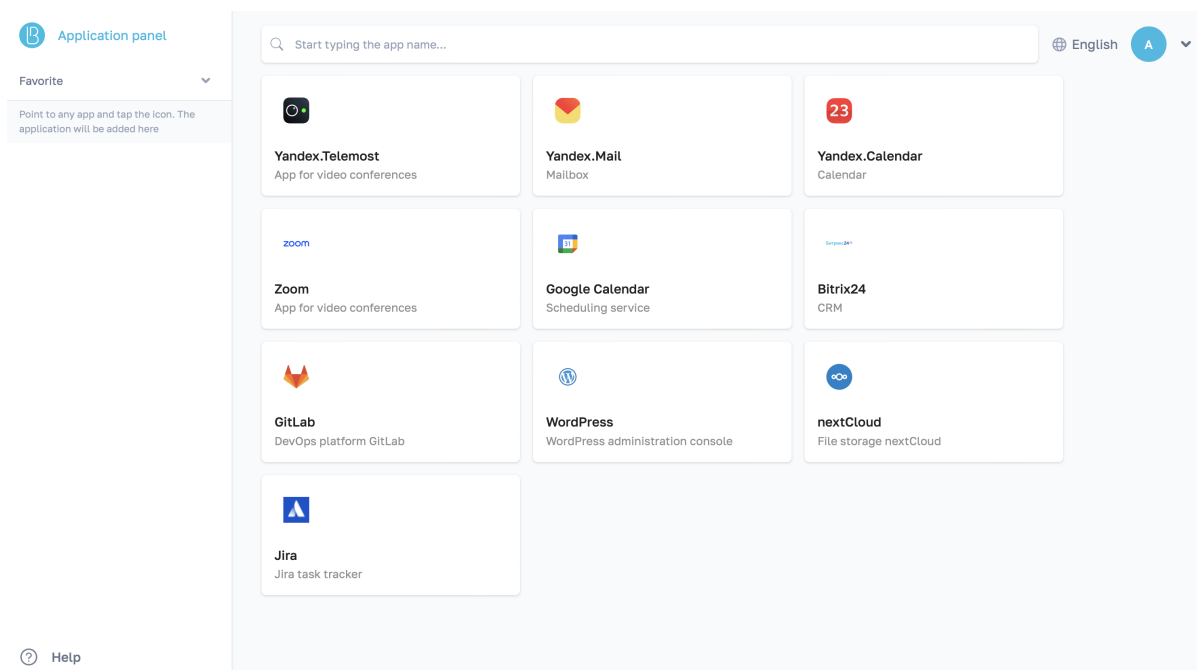
The operation of the `blitz-keeper` service is recorded into a separate log. To view the log, open the `blitz-keeper.log` file in the `/var/log/identityblitz/` directory.

```
sudo vim /var/log/identityblitz/blitz-keeper.log
```

## 4.2 Blitz Panel app showcase

### 4.2.1 About Blitz Panel

The Blitz Panel module is used for creating a panel that provides users with quick access to connected applications. Users can select the panel language, add frequently used applications to Favorites, and navigate to the User profile.



Administration of the module is performed via the `blitz-panel` service.

## 4.2.2 Installing the blitz-panel service

---

**Important:** See the [system requirements](#) (page 10).

---

To install the `blitz-panel` service, use the `blitz-panel.bin` installer.

---

**Important:** The `blitz-panel` service can be installed on any server where the Blitz Identity Provider server is installed.

---

To install `blitz-panel`, do the following:

1. Copy the `blitz-panel.bin` file from the Blitz Panel distribution to any directory on the designated server (for example, `/tmp`).
2. Run the `blitz-panel.bin` installer, specifying the `-j` launch parameter as `JAVA_HOME`, the JDK installation directory.

Blitz Panel will be installed in the `/usr/share/identityblitz/blitz-panel` directory.

```
cd /tmp
chmod +x blitz-panel.bin
./blitz-panel.bin -- -j <JAVA_HOME>
```

3. Create a `panel.conf` file with the initial Blitz Panel settings:
  - `IDP_DOMAIN` – the name of the domain with running Blitz Identity Provider;
  - `CLIENT_ID` – identifier for connecting the Blitz Panel application to Blitz Identity Provider via OAuth 2.0.

**Attention:** It's not allowed to use colons and tildes in `client_id`.

- `CLIENT_SECRET` – secret key for connecting the Blitz Panel application to Blitz Identity Provider via OAuth 2.0.
- `PANEL_DOMAIN` – the name of the domain on which Blitz Panel will be running.

Listing 1: Configuration file example

```
IDP_DOMAIN=mydomain.com
CLIENT_ID=qwerty12345
CLIENT_SECRET=54321ytrewq
PANEL_DOMAIN=mydomain.com/panel
```

4. Run the Blitz Panel initial setup script, specifying the path to the `panel.conf` file.

```
/usr/share/identityblitz/blitz-panel/bin/configure -f /tmp/panel.conf
```

As a result of the script execution, the Blitz Panel configuration files will be prepared.

### 4.2.3 Blitz Panel configuration

To configure Blitz Panel, do the following:

1. Put application icons into the `/usr/share/identityblitz/blitz-panel/static/resources/icons/` directory.

**Note:** The following formats are supported:

- SVG,
- PNG, maximum 128px on the minimum side.

2. In Blitz Identity Provider, [create](#) (page 171) an application to connect Blitz Panel to Blitz Identity Provider via OAuth 2.0.

**Application settings**

**Identifier (entityID or client\_id)**   
Application identifier. Used for identifying application within the SAML (corresponds to entityID) and OAuth 2.0 (corresponds to client\_id) protocols.

**Name**   
Human-readable application name. Is used only inside Blitz Identity Provider.

**Domain**   
Usually a link to the application's start page, e.g. http://testdomain.ru/. If TLS-authentication is used, then the domain should correspond to the domain specified in the certificate.

**Application start page**   
Link to the application start page, e.g. http://testdomain.com/private. When logging in using SAML, it is used as a link to go to the application in case the login page is opened from the browser history

**Identifier encryption key**   
If the key is specified, the user ID for the application will be encrypted using this key. The key value can be selected from a list. You can also assign a new key by typing it in the search box and pressing Enter

**Page template**   
Page template determines the login page appearance. If the template is not specified, then the default template is used.

**Application tags**   
Позволяют помечать приложения определенными признаками. И использовать их при настройке логики работы с данным приложением, например, анализировать в процедуре входа

Specify `client_id` and `client_secret` set during the Blitz Panel installation.

## Interaction settings

Secret (client\_secret)

.....



Application's secret (client\_secret). If defined, this secret should be used by the application when making a request to Blitz Identity Provider

Extra secret (client\_secret)

Enter application's extra secret for authentication



Application's extra secret (client\_secret). If defined, this extra secret should be used by the application when making a request to Blitz Identity Provider

Predefined redirect uri  
(redirect\_uri)

Enter predefined redirect uri

URL used for user redirection after successful authorization (redirect\_uri)

Redirect uri prefixes

To add a new prefix enter it and press Enter

A prefix is used to check the redirect uri. If the authorization request includes an redirect uri that doesn't correspond to any prefix, then the authentication is rejected

Available scopes

openid  
profile

The scopes that will be available to the application.

Default scopes

Scopes that are granted by default after authorization. If there are no default scopes, then the request must explicitly include the required scopes.

Be aware that you need to copy the URLs of API requests from the Protocols section to the `/etc/blitz-panel/app.conf` configuration file (see the next step).

## Protocols

SAML

OAuth 2.0

Simple

REST

RADIUS

For correct integration specify these links in the application settings

URL for authorization

`/blitz/oauth/ae`

The request for authorization should be sent to this URL (authorization endpoint)

URL to get and refresh a token

`/blitz/oauth/te`

The request for getting and refreshing an access token should be sent to this URL (token endpoint)

**Tip:** The values of `client_id` and `client_secret` set during the Blitz Panel installation can be changed if necessary in the same configuration file.

3. Open the `/etc/blitz-panel/app.conf` configuration file. In the `session -> oauth` section, set parameters to connect the Blitz Panel application to Blitz Identity Provider via OAuth 2.0.

- name: arbitrary connection name;

- `clientId`: check if the `client_id` application identifier matches the one specified in Blitz Identity Provider.
- `clientSecret`: check if the application secret key matches the one specified in Blitz Identity Provider.
- `logoutUrl`: URL that will be used by Blitz Panel to send the logout request to Blitz Identity Provider.
- `authUrl`: URL that will be used by Blitz Panel to send the user authorization request to Blitz Identity Provider.
- `tokenUrl`: URL that will be used by Blitz Panel to send the request to Blitz Identity Provider to obtain or update an access token.
- `me`: URL (`url`) that will be used by Blitz Panel to send the request to Blitz Identity Provider to receive a user data, and an attribute (`subjectIdAttr`) to search for a user in the storage.
- `scopes`: list of permissions that will be available to Blitz Panel.

```
"session": {
  "oauth": {
    "name": "Blitz IdP",
    "clientId": "CHANGE_CLIENT_ID",
    "clientSecret": "CHANGE_CLIENT_SECRET",
    "logoutUrl": "https://CHANGE_IDP_DOMAIN/blitz/login/logout",
    "authUrl": "https://CHANGE_IDP_DOMAIN/blitz/oauth/ae",
    "tokenUrl": "https://CHANGE_IDP_DOMAIN/blitz/oauth/te",
    "me": {
      "url": "https://CHANGE_IDP_DOMAIN/blitz/oauth/me",
      "subjectIdAttr": "sub"
    },
  },
  "scopes": [
    "openid",
    "profile"
  ]
},
...
},
...
```

4. If necessary, set a user session parameters: the URL to which the user will be redirected after logging out, the TTL value, the maximum period of inactivity in seconds, the period between the session activity checks in milliseconds, the created `cookie` name, etc.

```
"session": {
  ...
  "postLogoutUrl": "/blitz/panel",
  "ttlInSec": 36000,
  "inactivityPeriodInSec": 3600,
  "checkSessionPeriodInMs": 1000,
  "cookie": {
    "name": "scs",
    "path": "/blitz/panel",
    "transient": true
  },
  "useCompression": false,
  "encodingKey": "CHANGE_SCS_ENC",
  "hmacKey": "CHANGE_SCS_HMAC"
},
...
```

5. The `apps -> sources` section contains groups of applications that can be formed according to arbitrary characteristics (static, dynamic, etc.). Each group has a name, a list of applications in the group, and rules that determine which users the applications are shown to.



In the `apps -> sources -> rules` section, set the rules that determine for which users certain applications will be displayed.

Each rule consists of the following parts:

- `name`: the rule name.
- `conditions`: conditions for user selection.

The following types of conditions are supported:

- `"typ": "userGroup"` - *user group* (page 149). You must specify the group profile name and group ID.
- `"typ": "userClaims"` - flexible selection of users based on the claims regarding their attributes. A condition of this type can contain statements on multiple attributes. In order for a user to be selected according to a condition, they must satisfy **all** statements in it.

**Attention:** A rule can contain multiple conditions. A rule is applied to a user if the user meets **at least one** condition.

- `tags`: tags linking user selection rules and applications.

The following types of tags are supported:

- arbitrary parameter (for example, `role`, `department`, etc.);
- application identifier (set in the `appId` list).

**Attention:** A rule is applied to an application if at least one of the values specified in this section is present in the application settings (see the next step).

6. In the `apps -> sources -> apps` section, set the list of applications connected to Blitz Identity Provider, that will be displayed on the panel. For each application, specify the following settings:

- `id`: application ID in Blitz Identity Provider.
- `name`: the name of the application that will be displayed on the panel, in required languages.
- `url`: URL of the application's start page.
- `icon`: the icon file name in the `/usr/share/identityblitz/blitz-panel/static/resources/icons/` directory.
- `tags`: tags that determine for which users the application will be displayed on the panel according to the rules specified above (optional).
- `desc`: description of the application in required languages.

Listing 2: Example of setting up rules and application list

```
"apps": {
  "sources": [
    {
      "name": "Static Applications",
      "type": "static",
      "apps": [
        {
          "id": "dev_portal",
          "name": {
            "en": "Developer Tools 24"
          },
          "url": "https://my.domain.com/dev/portal",
```

(continues on next page)

(continued from previous page)

```

        "icon": "confluence.svg",
        "tags": {
            "role": [
                "admin",
                "sys_admin"
            ]
        }
    },
    {
        "id": "jira",
        "url": "https://my.domain.com/dev/jira",
        "name": {
            "ru": "Jira"
        },
        "icon": "jira.svg",
        "tags": {
            "role": [
                "admin",
                "sys_admin"
            ]
        }
    },
    {
        "id": "test-app",
        "url": "https://my.domain.com/dev/test",
        "name": {
            "en": "Test application"
        }
    },
    {
        "id": "atom",
        "url": "https://my.domain.com/dev/atom",
        "name": {
            "en": "Atom"
        },
        "desc": {
            "en": "Atom is your essential companion"
        }
    },
    {
        "id": "call_center",
        "url": "https://my.domain.com/dev/call",
        "name": {
            "en": "Call center"
        },
        "desc": {
            "en": "Call center management application"
        },
        "tags": {
            "role": [
                "admin",
                "sys_admin"
            ]
        }
    },
    {
        "id": "web_mail",
        "name": {
            "en": "Mailbox"
        },
        "desc": {

```

(continues on next page)

(continued from previous page)

```

        "en": "Corporate mailbox"
    },
    "icon": "gmail.svg",
    "url": "https://my.domain.com/dev/portal",
    "tags": {
        "role": [
            "sys_admin"
        ]
    }
},
{
    "id": "yandex",
    "url": "https://my.domain.com/dev/yandex",
    "name": {
        "en": "Search engine"
    },
    "desc": {
        "en": "Search engine web interface"
    }
}
],
"rules": [
    {
        "name": "admin_role",
        "conditions": [
            {
                "typ": "userGroup",
                "profile": "main_group_profile",
                "id": "app_admin"
            },
            {
                "typ": "userClaims",
                "claims": {
                    "company_type": "IT",
                    "position": [
                        "head",
                        "master"
                    ]
                }
            },
            {
                "typ": "userClaims",
                "claims": {
                    "company_name": "Моя компания"
                }
            }
        ],
        "tags": {
            "appId": [
                "dev_portal",
                "yandex"
            ],
            "role": [
                "admin",
                "sys_admin"
            ]
        }
    },
    {
        "name": "atom",
        "conditions": [

```

(continues on next page)

(continued from previous page)

```

        {
            "type": "userClaims",
            "claims": {
                "tags": [
                    "atom"
                ]
            }
        },
        {
            "tags": {
                "appId": [
                    "atom"
                ]
            }
        }
    ]
}
},
...

```

7. Add the `blitz-panel` service to autostart and launch it:

```

systemctl enable blitz-panel
systemctl start blitz-panel

```

#### 4.2.4 Blitz Panel design and localization

##### Appearance modification

If necessary, you can change the appearance of the panel by making changes to the files in the `/usr/share/identityblitz/blitz-panel/static` directory. You can customize the following elements:

- favicon;
- the `index.html` template;
- CSS styles (`../resources/styles.css`).

##### Adding a language

To add a language, put the file with translated strings `<two-letter language code>.json` (for example, `ar.json` for Arabic) into the `/usr/share/identityblitz/blitz-panel/static/resources/locales` directory and restart the `blitz-panel` service.

```

sudo systemctl restart blitz-panel

```

The new language will appear in the language selection menu of Blitz Panel.

**Note:** An application name and description seen on the panel are *localized* (page 456) via the `/etc/blitz-panel/app.conf` file.

### 4.2.5 Viewing logs

The operation of the `blitz-panel` service is recorded into a separate log. To view the log, open the `blitz-panel.log` file in the `/var/log/identityblitz/` directory.

```
sudo vim /var/log/identityblitz/blitz-panel.log
```